Aalto University
School of Business

TIMEOUT ANALYSIS IN REAL-TIME BIDDING FOR AN ADTECH COMPANY

Master's Thesis
Yen Hoang
Aalto University School of Business
Information and Service Management
Fall 2020

Aalto University
School of Business

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Yen Hoang

Title of thesis TIMEOUT ANALYSIS IN REAL-TIME BIDDING FOR AD-TECH COMPANY

Degree Master of Science in Economics and Business Administration

Degree programme Information and Service Management

Thesis advisor(s) Timo Kuosmanen

Year of approval 2020 **Number of pages** 50 **Language** English

Abstract

Real-time bidding is the popular way to exchange online ads through instantaneous auctions. To ensure best user-experience and highest profitability, there need to be a time limit to prevent excessively slow auctions. That limit is called timeout value. The timeout value should be chosen carefully to balance the benefits of users, advertisers and publishers.

This study analyzes bidding data collected by an ad-tech company to determine which factors strongly affect the timeout result and whether those effects are positive or negative. Relevant academic papers in the field are scrutinized to find out promising factors which should be examined further. Data of those factors will be visualized in the descriptive analysis to give general ideas about the variables and their distribution. After that, random forest and logistic regression models are used to rank the factors and estimate their significance on bid timeout result.

The study finds out that number of networks, browsers and bid networks are categories that have strongly relevant factors. Some other categories like geo-location, device families and placement types also have certain relationship with the timeout result. Therefore, the paper suggests that timeout values should be set with high level of details, for example, per browser, per device family or per geo-location, to reflect the effects of those factors on timeout result.

Keywords real-time bidding, online advertising, timeout

Acknowledgements

I would like to thank my thesis supervisor, Timo Kuosmanen, who provided me with valuable feedbacks and guided me throughout the research process.

Thanks Dat and Kiosked for providing me with the deep insights and technical supports on the topics.

Finally, I am thankful to Lauri for his continuous encouragement, patience and support.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Background information

Starting with the AT&T's banner in HotWire magazine in 1984, online advertising has grown to become an unavoidable part of internet users' experience. At the moment, online advertising has developed to many different forms: displays, search, classifieds, e-mail, lead generation, interstitials, etc (Evans, 2009). Everyone who has used the internet in the last few years should have come across at least one of those forms. Along with the form of presentation, the way to exchange online advertisements also evolves through time. Starting from direct negotiation between advertisers and publishers, most of the online ads are now sold through a long chain of agencies with different specializations. Thanks to the development in technology, ads could be purchased through that long chain in around 1 second. The method is called real-time bidding.

Introduced in 2009, Real-time bidding (RTB), or programmatic buying, is the way to buy and sell online-advertisement through instantaneous auctions. According to Sayedi (2018), the value of the US real-time bidding market is more than 14 billion USD in 2017, accounting for over one third of the display advertising market in the United States. In 2020, real-time bidding is projected to occupy 86% of the online display advertising market in the country (Srinivasan, 2020).

In real-time bidding, the ads need to be auctioned and completely loaded during the time that users load and scroll through the sites. Even though the ads should be presented right when users are in the spot, the auctions cannot start too early due to the advertisers' preference. For example, advertisers do not want their ads to appear under the 20th paragraph while users are just reading the 1st paragraph of a long article. There is a chance that users could leave before reaching that point and the ads become useless. As a result, auctions for ads are only started when users are nearly in view, starting when users are in the 18th, 19th paragraph in the above example. To serve ads in such a short time period, the bids need to be collected very quickly. For many different reasons, the time it takes for bidders to submit their bids can be substantially different. Some bidders send bids in half a second, while others take several seconds or longer. To cope with that difference, there needs to be a time limit value. When a bidder fails to answer within that time limit and loses the chance to show ads, it is called timeout. For example, if the time limit is 2000ms, the auction will start when all bids are received or after 2000ms. In the later case, late bids which arrive after 2000ms will be considered as invalid and the winner will be chosen only between valid bids.

1.2 Business case

The client company analyzed in this thesis, Kiosked Ltd., is an ad-tech company who helps publishers do the auctions, connecting between them and the bidders, commonly called supply-side platforms (SSPs), by self-developed scripts. The scripts are unique for each website/device family/ad size/etc. They are placed and run in the website's header or through a third-party ad server. When triggered, they send the bid requests to all bidders in the stack.

The company manages around 200 websites from publishers all around the world with a wide range of characteristics that could affect timeout value. They want to get more insights into the latency of their current websites and find out suitable timeout values for each website.

The study aims at providing the client company with thorough insights about factors that influence latency of their script. From there, they could identify the likelihood that a script gets timeout to determine the correct timeout value range. At the moment, the timeout value for each script is still set at a fixed default in the system with occasional manual adjustment based on personal judgments and experience. The insights are expected to rank influential factors in terms of importance and separate whether they have positive or negative effects on the timeout status.

1.3 Research Objectives

The thesis analyzes Kiosked's dataset of bid events to find out the link between the bid events' characteristics and the bid results. Bid results are the string answers received from the networks for bid requests. The result could either be timeout or not timeout. The not time-out result could either be a bid or a no-bid decision but it indicates that the system got response from the networks before the time limit. There are 3 questions that need to be answered:

1. Which factors have significant influence on the bid result?
2. How important are those factors? How strong are the effects they have on the bid results?
3. Are the effects positive or negative? Or in other words, does the factor increase or decrease the chance that a bid gets timeout?

2 Literature Review

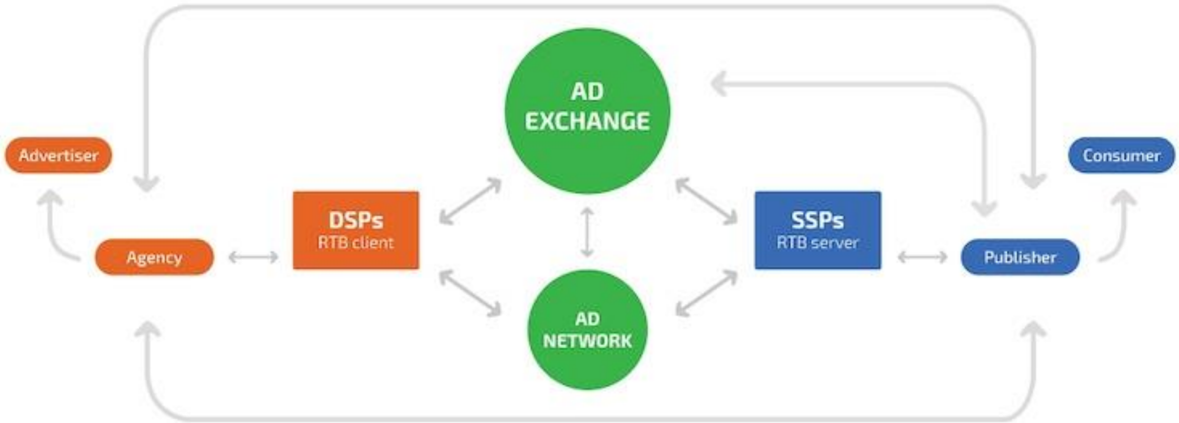

Figure 1: The positioning of ad networks and ad exchanges in the online advertising ecosystem (SelfAdvertiser, 2017)

The mechanism of RTB is illustrated in Figure 1. When the consumer visits a website, the publisher sends bids to different supply-side platforms (SSPs). The case company is the connection between publishers and SSPs in this step. Each SSP then checks with all of their demand-side platforms (DSPs) to come up with the most suitable, highest bid. SSPs send the bids back to the publisher to do auctions. SSP with the highest bid wins and the ad is displayed immediately on the slot.

The amount of time each SSP takes to return a bid (latency) varies significantly, depending on several factors such as the loading speed of the consumer (device & internet speed), loading speed of the site (how heavy the site is, the amount of ads/video/image the site has), responding speed of the SSP networks and technical errors. If the publisher waits for all networks to respond, the process may take too long. During that time, users may already scroll past the ad slot and the ad will never get shown. The timeout limit could help to reduce that waiting time and increase the chance that ads are in view. However, if the timeout value is too short, there may be no bid received or the auction missed higher value bids. As a result, it is important to understand which factors affect the bid latency to set an optimal timeout value for publishers to maximize revenue from real-time bidding.

As real-time bidding is a very new and rapidly changing area, there are not many up-to-date research papers about that topic, especially about timeout and latency. Existing research works find out several different factors that could affect latency of the bidding process by experiencing technical processes (Kumar, 2017), analyzing vast amounts of top websites' data with 35,000 top websites listed by Alexa (Pachilakis, et al., 2019), machine learning optimization (Dewitt, 2019) and tests on rendering/scripting engines (Nielson, et al., 2008).

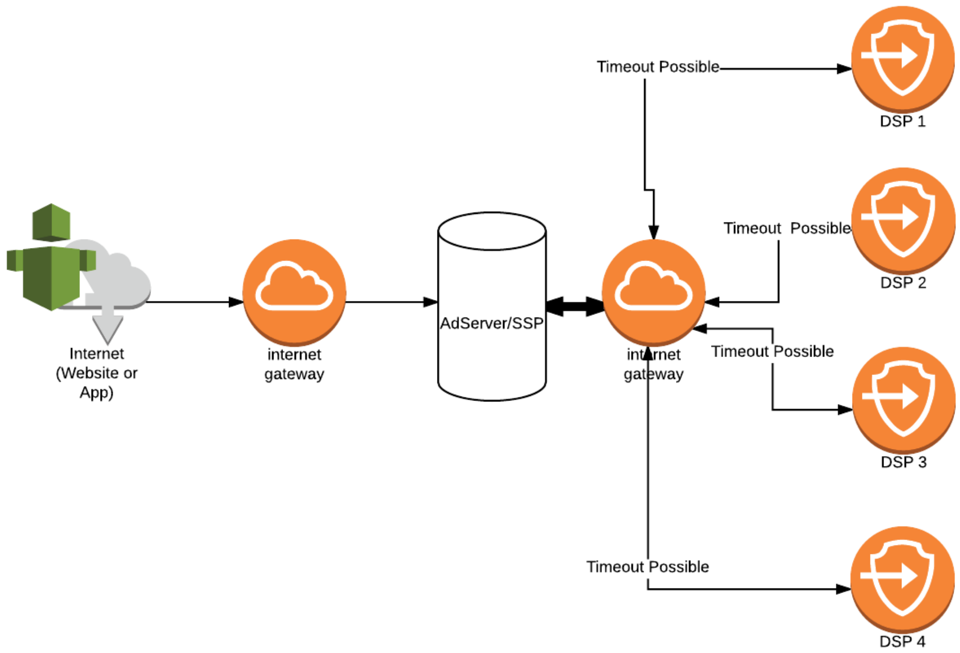

Figure 2: Flow Diagram of Ad request (Kumar, 2017)

From a technical point of view, the latency can come from the network between SSPs and DSPs (request transferring speed) or from within the DSPs (bidding speed) (Kumar, 2017). The flow of ad requests is demonstrated in the Figure 2. According to the paper, the issue could arise from the connection between the SSP/Ad server and data center of a certain demand partner. So each demand partner could have different connection speed and a different chance for errors to arise. Not to mention, the time to process the bid and come up with the response within the DSP is very specific for each DSP (ibid). As a result, the paper gives us suggestions that the type of DSPs participating in the bid could affect bid latency and bid result. Pachilakis, et al., (2019) agree with that suggestion. They claimed that there are differences in technological levels between networks. Some networks are still in the transition stage from the previous technology(waterfalling), which slow down bid responses in comparison with networks using new technology (Header bidding). In addition, the research also pointed out that some networks can take significantly longer than others to answer (ibid).

Beside the type of networks, Pachilakis, et al., (2019) also suggested that there is a link between the number of demand partners and the bid latency as websites' loading speed could be slowed down by the increasing number of demand networks. It is worthy to note that in that research, half of the analyzed websites only use 1 demand partner while 20% work with at least 5 and only 5% have more than 10 partners (ibid). We can compare this ratio with Kiosked's ratio later when analyzing the link (if any) between the number of demand partners and the bid result.

Apart from the above factors, device family, geo location and network condition are suggested as vital predictors of bid latency and bid timeout by the introduction of Index Exchange's adaptive timeout innovation (Dewitt, 2019). Index Exchange is one of the major demand partners in the US. The innovation used machine learning to set timeout values for the script based on the users' device family, geo location and the network latency in each page view, instead of setting the fixed, general timeout values (ibid). The product solves exactly the issue we want to tackle in the thesis. Unfortunately, as this is a commercial product, the detailed research and models behind it are not revealed publicly but we can consider the influential factors suggested by them. According to the article, the adaptive timeout setting is effective in reducing the timeout rates in many different test cases (ibid).

Last but not least, Nielson, et al. (2008) made extensive tests over the web performance on different browsers. The paper mentions several rendering and scripting engines but, as Kiosked's scripts are delivered as Javascript tags, I only consider the result regarding Javascript performance. There is a significant difference in the rendering speed on different browsers with Opera and Safari are the fastest (Nielson, et al., 2008). As a result, I expect to see low timeout rates on bids going through Safari and Opera browsers, compared to bids on other browsers. We can check this hypothesis by analyzing the data and looking into the model's result.

In summary, we aim to find out the factors that influence the bids' result (timeout/not timeout). With a fixed timeout value, bid latency is the key determinator of bids' results. According to the literature review, the following four factors influence the bid's latency: amount of demand partners (Pachilakis, et al., 2019), types of demand partners (Kumar, 2017; Pachilakis, et al., 2019), device family, geo location and speed of connection (Dewitt, 2019). The speed of connection is affected by the browser (Nielson, et al., 2008).

As a result, our hypothesis is that bids' result is influenced by the number of participating networks, bid networks, device family, geographic location (country) and browsers. In addition, according to Kiosked's data (Figure 12. below), there is also a difference in timeout rates among different placement types so that factor is taken into consideration. Placement-type is an internally defined category, based on the in-house developed configuration so it is not possible to find available academic research to support it.

Also according to the analyzed literature, the number of networks is supposed to have negative effects on the bids' result, or in other words, the increasing number of networks could lead to higher timeout rate. Meanwhile, browsers Safari and Opera are expected to have positive influence by rendering the script faster.

3 Data exploration

3.1 Data source

The thesis utilizes first party data collected by Kiosked Ltd. The data system is built on Amazon AWS (Amazon Web Services) and can be analyzed in real time. The raw complete data is stored for 6 months. The data used in this thesis is taken from two tables: facts.prebid_full and facts.prebid_bid. None of the data used relates to, or can be used to identify, a natural person. As a result, the thesis does not use any personal data according to the GDPR's definition (European Union, 2016).

The data at Kiosked is collected when the company's scripts run on the publishers' websites. Only 10% of the live data is recorded due to storage limitations but the estimation made based on those data for revenue, CPM, RPM, etc has very small error compared to actual data provided by network partners. With that sample rate, there is data of around 2.5 billion bids from 274 million auctions available to examine.

Due to the computer's processing limitation, only 50,000 events (containing 300,000-400,000 bids) are included in each dataset. There are 3 datasets analysed so 150,000 different events are randomly chosen. Since the timeout values set for each script could directly influence the completion of the bid, the timeout value in each model needs to be the same to evaluate the influence of other factors. Unfortunately, the timeout values are already set by the company and cannot be adjusted to the one single value. To deal with that issue, I consider 3 datasets with each set containing only bids with the same timeout values. The three values chosen are 1800, 1500 and 800. They are the top 3 most common timeout values set in Kiosked's bids (Figure 3). In addition, as the Kiosked's timeout values range from 700ms (shortest) to 2000ms (longest), the 3 values mentioned above also evenly represent the full range from short (800ms) to medium (1500ms) and long (1800ms).

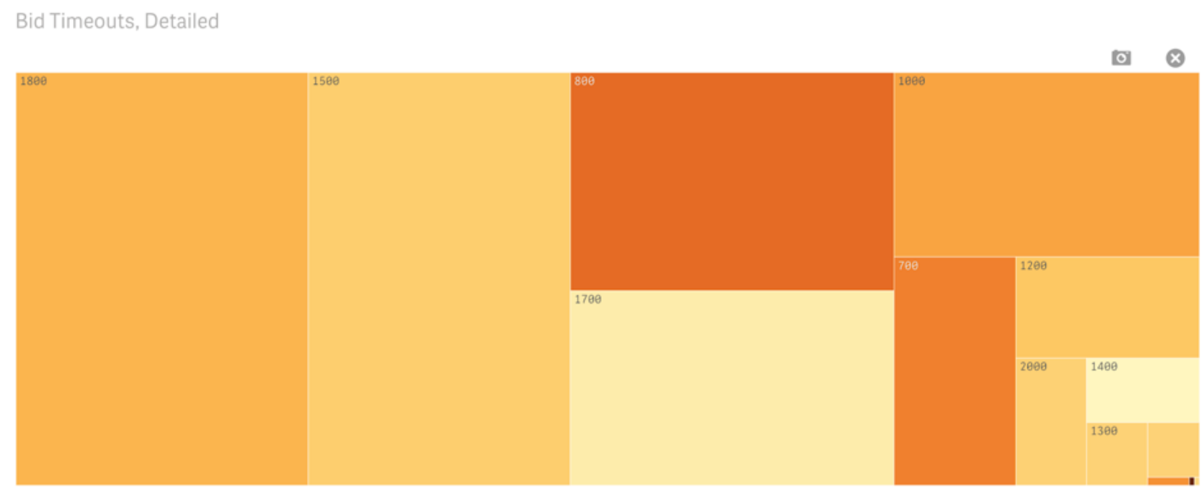

Figure 3: Distribution of Header-bidding timeout values in Kiosked's bids (size of the box corresponds to the amount of bids run with the given timeout values)

3.2 Variables

Table 1 introduces the data utilized in this thesis along with description, data types and examples to familiarize the readers with the common terms and variables that will be analyzed henceforth.

Table 1: Variables explanation

Column names	Description	Data type	Example
'type'	Type of the event in the auction	string	pbStart, pbEnd
'event_id'	An event identifier that should be unique across all events generated in the client script	string	43bc9511bf70ab93d2c404d8877583016b2
'browser'	The browser of the user	string	Chrome, Safari
'country'	The user's country location at the time of connection	string	US, JP
'device_family'	The type of device that user used to	string	Desktop,

	browse the website		tablet, mobile
'placement_type'	The type of placement shown according to Kiosked's definition	string	In-screen, in-line
'no_networks'	Number of networks participating in the auction	Positive integer	4,5,6
'bid_network'	Name of the network that sends the bid	string	AOL, openx
'bid_result'	Answer from the networks for bid requests	string	b, n, t

One single auction includes many different bid requests, each request for a participating network. A complete bid request has maximum 5 event types ('type' variable values): pbStart, pbHbStart, pbHbEnd, pbEnd and pbHbLate. There are pbStart/pbEnd and pbHbStart/pbHbEnd to mark the difference between header bidding (HB) steps and Google's DoubleClick for Publisher (DFP) step. The DFP step could be before or after the HB steps but usually it is after. The time sequence of the steps is as follow:

pbStart: mark the start of the auction

⇨ pbHbStart: the request is sent to each HB partner

⇨ pbHbEnd: receive the answer from the HB partner

⇨ pbEnd: same information as in pbHbEnd but include DFP's response

⇨ pbHbLate: If the bid comes after the time limit, it is still recorded (amount, time of arrival, network, etc) but is not considered in the auction.

The event_id values are unique across all events generated and are the identifiers of those events. Entries of the same type in the same auction have the same event_id. For example, auction A has 9 participating networks then there will be 9 entries of type 'pbStart' with the same event_id 'abcd1234', 9 entries of type 'pbEnd' with the same event_id 'efgh5678' and so on.

The 'browser','device_family' and 'country' variables refer to the browser and device that the users use when browsing that ad slot, as well as the country where the user is connected from.

The 'placement_type' variable categorizes ads based on the configuration and some of the ads' characteristics. There are 4 placement types: static, in-line, in-screen and multi. 'Static' placement is run inside the publishers' containers while all others are run inside Kiosked's containers. In-screen placement is sticky to one edge of the website (usually bottom edge) and remains visible when the users scrolling around. In-line placement is attached to one element in the website, for example under the images, between the paragraphs or between the side boxes. In-line placement will be out of view when the users scroll away from the ads' spots. Finally, multi placement is mostly the same as in-line placement but it includes 2 ads next to each other. Those ads are mostly 300x250 sizes.

The 'bid_network' variable records the name of the demand partner that gets the bid requests. The 'no_networks' variable is the number of demand partners participating in a particular auction. The variable counts the bid networks from the start so no matter if the requests get response or if there is any error in the process, the network is still counted.

The 'bid_result' variable indicates the response from the networks to the bid requests. The response could be either a bid (recorded as 'b'), a denial to bid (recorded as 'n') or no timely response/timeout (recorded as 't').

3.3 Data preparation

As mentioned above, the data utilized for the model is taken from two separate tables in the database so they need to be prepared and merged before going into the model.

First of all, the two data tables imported from SQL will be merged based on the event_id key. The event_id is unique for each row in the prebid_full table while there are many rows with the same event_id in the prebid_bid table. As a result, I merge prebid_full table into prebid_bid table so that we can keep all the bid rows. The models only use 'pbEnd' type events of the bids since they include all necessary information about the bids' characteristics and are unique for all finished bids. We do not consider unfinished bids in this thesis since they happen when users scroll away before we get the bids' result, not because of the timeout setting.

Figure 4 shows the sample of the data that will be processed in this section. First column is the index numbers of the rows, starting from 0. That number only shows the position of the rows in the table. As the rows are long, the display is divided to 2 parts with 4 columns in each half. Lines with the same indexes belong to the same row. The first row is the labels of columns included in the data, the index column does not have label. There are 8 informative columns in the table (excluding index column). For example, the first row has index = 0, event_id = '01ek91zwadn49r10hh1kw0765', bid_network = 'criteo_premium', bid_result = 'n', type = 'pbEnd', browser = 'Chrome', country = 'US', device_family = 'desktop' and placement_type = 'in-screen'.

	event_id	bid_network	bid_result	type
0	01ek91zwadnp49r10hh1kw0765	criteo_premium	n	pbEnd
1	01ek91zwadnp49r10hh1kw0765	ix	n	pbEnd
2	01ek92qysqdafzprv1r4sg8esn	appnexus	n	pbEnd
3	01ek92qysqdafzprv1r4sg8esn	appnexus	b	pbEnd
4	01ek94wjvxekg421wfsccbddc8	criteo_premium	b	pbEnd

	browser	country	device_family	placement_type
0	Chrome	US	desktop	in-screen
1	Chrome	US	desktop	in-screen
2	Chrome	US	desktop	in-screen
3	Chrome	US	desktop	in-screen
4	Chrome Mobile	US	mobile	in-screen

Figure 4: Sample data table after merging (first 5 rows)

Next step is adding the dependent variable to the data table. We need to modify the bid_result column from categorical values: bid (b), no bid (n) and timeout (t) to binary values. As the timeout status is the aim of the model, timeout (t) value is replaced by 0 while bid (b) and no bid (n) values are replaced by 1.

Some columns in the table contain many minor categories, with only a few rows for each category, that are not meaningful to examine separately. To deal with that issue, I group several minor categories into one to reduce the amount of dummy variables that need to be created and increase the significance of those categories. There are 2 columns that need to be prepared: browsers and country

With the browser column, minor versions of popular browsers will be combined to the same group with the popular ones. For example, 'Firefox iOS', 'Firefox Mobile', 'Firefox Beta', 'Pale Moon (Firefox Variant) are changed to ‘Firefox’ group; 'Chrome Mobile', 'Chrome Mobile iOS', 'Googlebot' are changed to ‘Chrome’ group. Smaller browsers those are unrelated to any big category like 'Amazon Silk', 'Opera', 'AppleMail', 'Pinterest', 'PhantomJS', 'Yandex Browser', 'MicroAdBot', 'QQ Browser Mobile', etc will be combined to group ‘Others. Even though the last group is the combination of many categories, it occupies only a small percentage of table’s data since each category only has a few or dozens of rows.

With the country columns, there is traffic from hundreds of countries. First, the countries are grouped together by continent: South America, North America, Europe and Asia-Pacific (including Oceania). There are only 2 exceptions that need to be modified to provide better insights:

The US is a major traffic source where about 40% of Kiosked's traffic comes from. According to its importance, US traffic is placed into a separate group. The rest of North America traffic is left in the NA (North America) group.

In Asia pacific region, traffic from Japan, HongKong, Australia and South Korea usually has much higher performance (CPM, RPM, revenue) than the rest of Asian traffic (i.e South East Asian, Indian, Middle East, etc). Those are also developed countries with better infrastructure than other Asian countries so internet speed is expected to be better and timeout chance is expected to be lower. As a result, Japan, Hong Kong, Australia and South Korea are placed into the P_AS (Premium Asia_ group while the rest of Asian traffic stays in the AS group.

To utilize categorical data in the model, they need to be converted into dummy variables with 0/1 values instead of string values. The columns that need to be converted to dummy variables are 'browser', 'device_family', 'placement_type', 'country' and bid_network. After dummy variables are created, the original columns are removed, together with one dummy variable for each group. The removal of one dummy variable for each group is to avoid multi-collinearity problems in statistics where 2 or several explanatory variables in a regression model are linearly related. To be specific, if we do not do the removal, the sum of all dummy variables of each group are always equal to 1, which makes them have a perfect linear relationship with each other. The variables to be removed are the largest ones in the group as they can be viewed as the reference. Those chosen variables are browser_Chrome, device_family_desktop, placement_type_in-line, country2_US and bid_network_appnexus.

When the table columns are well-prepared, we split it into train/test sets with the 70/30 ratio, 70% is the training set and 30% is the test set. The data sets are highly imbalanced with the majority of the bids being not timeout (see Figure 3. below) so we need to rebalance the sets by Python's imblearn package. According to Chawla, et al. (2002), the combination of oversampling the minority category and under-sampling the majority category gives better performance than solely under-sampling or oversampling. The article also suggests that the under-sampling range between 50% and 125% is a good range as it gives better results than other methods considered. Since we have a heavy dataset of over 300,000 bids, I decided to focus more on under-sampling rather than oversampling to reduce the processing burden when running the model later. As a result, the dataset is first oversampled with a sampling target of 20% to increase the amount of timeout bids to 20% of non-timeout bids. Next, the data is under-sampled with the rate of 100% to reduce the amount of non-timeout bids to equal to the amount of timeout ones.

4 Descriptive analysis

Before getting into modelling, all categories in the dataset are visualized to give the readers more insights about the possible variables and their distributions. We will discuss two types of descriptive analyses for each category: checking the distribution of individual variables in the same category and analyzing the timeout percentage of each variable. From the former type of analysis, we can detect the popular variables and outliers in each dataset as well as having a general view over what kind of bids each dataset has. Meanwhile, from the latter, we expect to have a preliminary estimation of each variable's relationship with the bid result. The result in this step tests the hypotheses suggested by the literature review and is the comparison point for the result in the significant test later on.

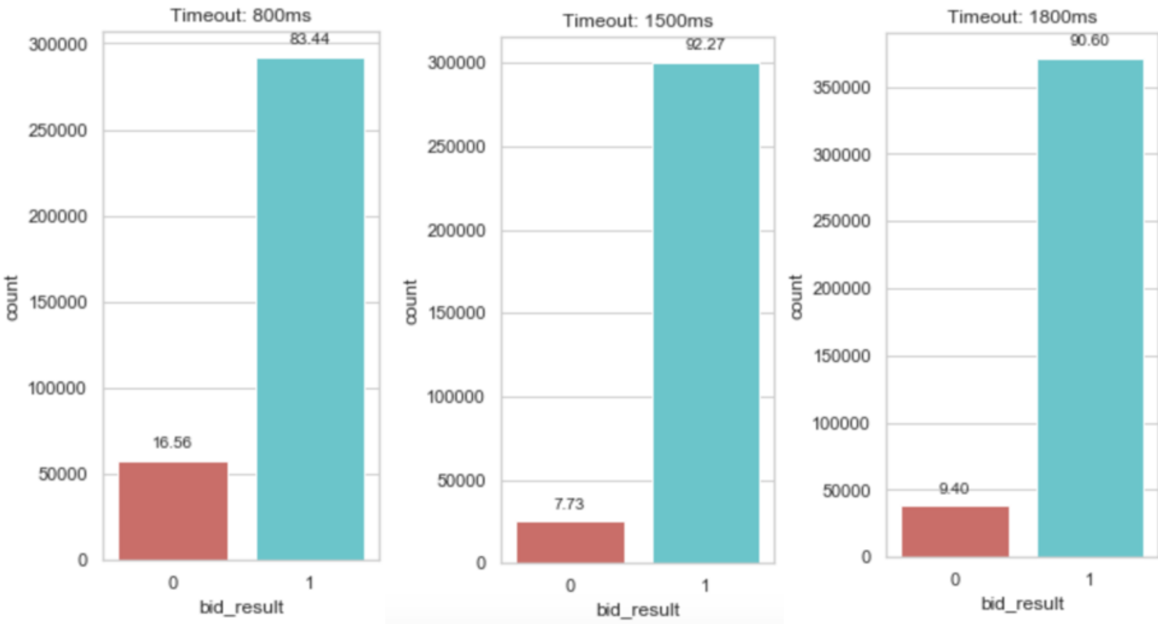

Figure 5: Bids' timeout rate by percentage (%)

Figure 5 illustrates the timeout rate of sample bids in the 3 datasets. As can be seen, timeout rate ("0" value) only occupies for a small fraction of total auctions. To be specific, it makes up for 16.56%, 7.73% and 9.4% of the total bids, respectively for timeout values of 800ms, 1500ms and 1800ms. The chart shows that the bids are much more likely to timeout when the timeout setting value is short (800ms). The difference in case of 1500ms and 1800ms is less noticeable.

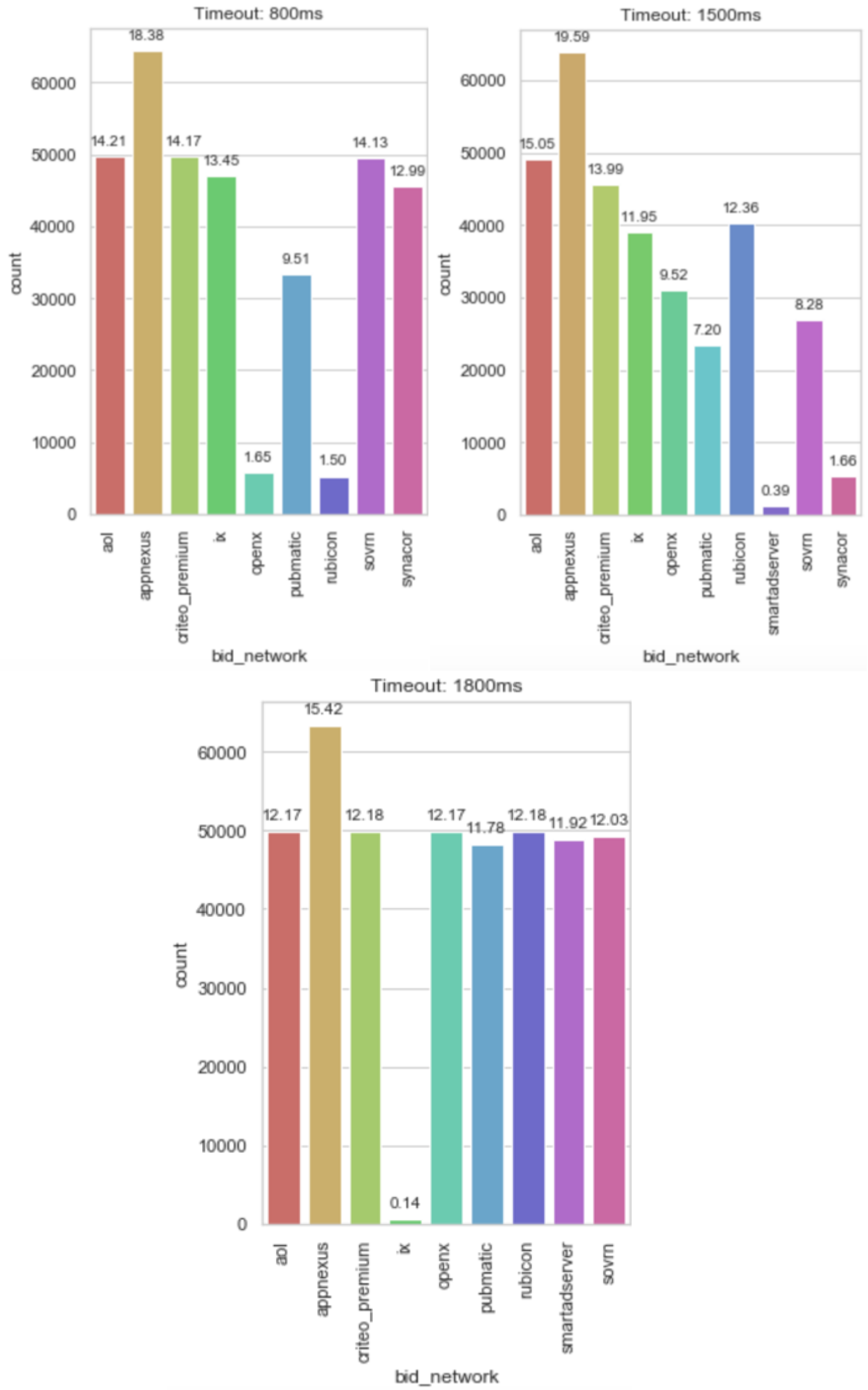

Figure 6: Bid networks distribution by percentage (%)

Figure 6 shows that in the timeout 1800ms set, the networks (only except for ix) are evenly distributed while in the other 2 sets, the networks' popularity varies greatly. Appnexus is the most popular network in all the cases while ix is only strong in the 800ms set and smartadserver is only strong in the 1800ms set. The reason behind that difference is just about how often the network receives the requests, not about the timeout setting. Some networks are more popular in certain geographic areas and those areas are usually set with certain timeout values. For example, ix (Index Exchange) is most popular in the US and most US sites are set at 800ms timeout. Figure 7 below shows the country distribution among bids with 800ms timeout value in the whole Kiosked's dataset.

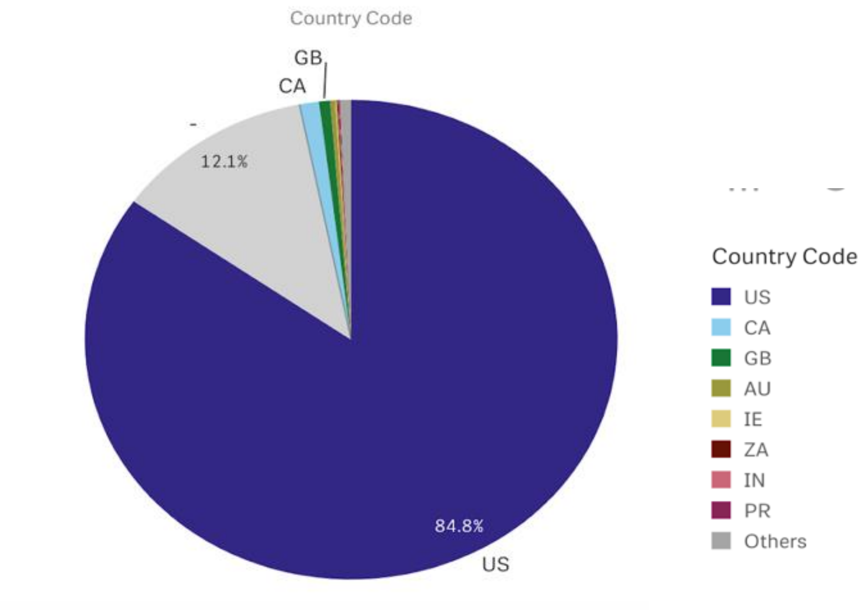

Figure 7: Country distribution among Kiosked total bids for Timeout 800ms

The networks distribution above only reveals about characteristics of the bids in the dataset but not much about how networks' timeout situation is. The figure 8 below is going to give more insights about how likely are the bids' timeout for each network.

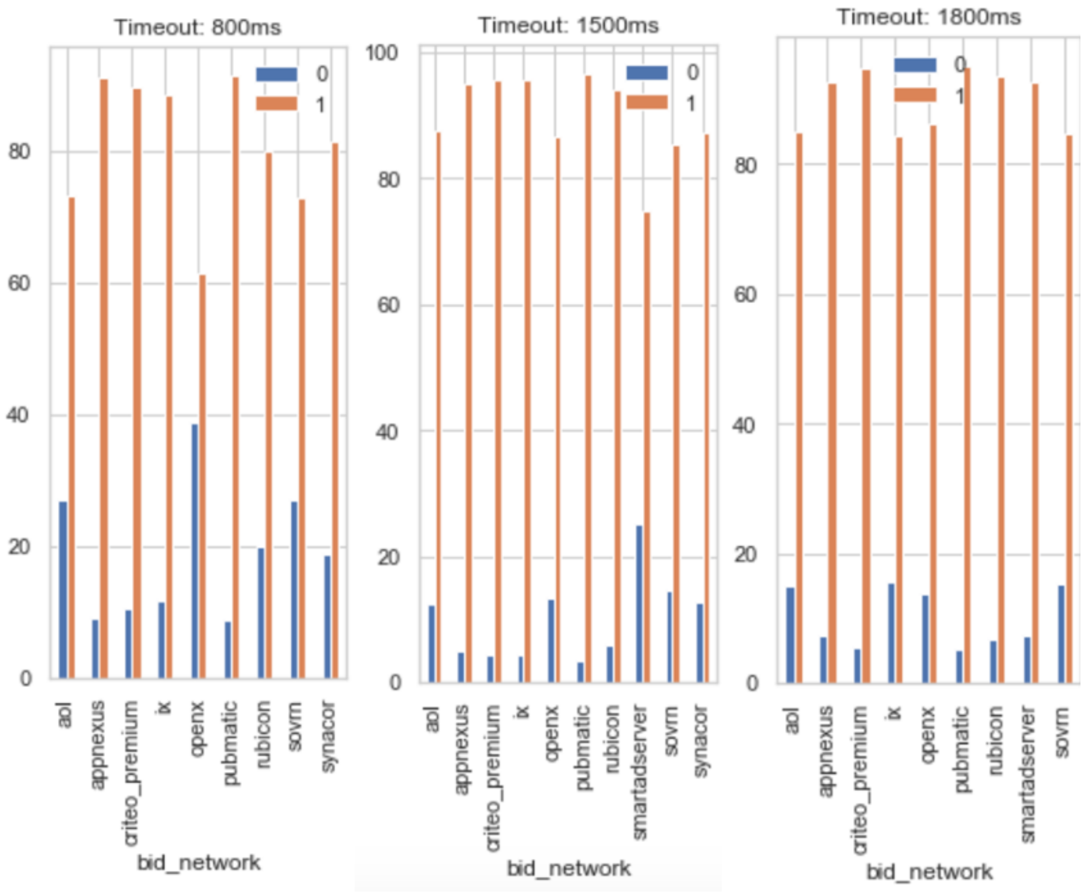

Figure 8: Time-Out percentage by networks

The figure 8 shows that the overall timeout rate is higher in the 800ms set compared to the 1500ms and 1800ms sets but the level of each network to the timeout change differs significantly. Appnexus, criteo_premium and synacor see a small decrease in timeout rate when the timeout values increase while timeout rates of openx, aol, sovrn and smartadserver drop to half. Ix (Index Exchange) is the interesting case when its timeout rate at 1800ms set is higher than in 800ms set and 1500ms. It’s likely due to the small sample size that index exchange has in the 1800ms set, according to Bid networks distribution in figure 6. Generally, the networks can be divided into 2 groups based on timeout rate. The high-timeout rate group includes aol, openx, sovrn and synacor while the low-timeout rate group has appnexus, criteo_premium, ix and pubmatic.

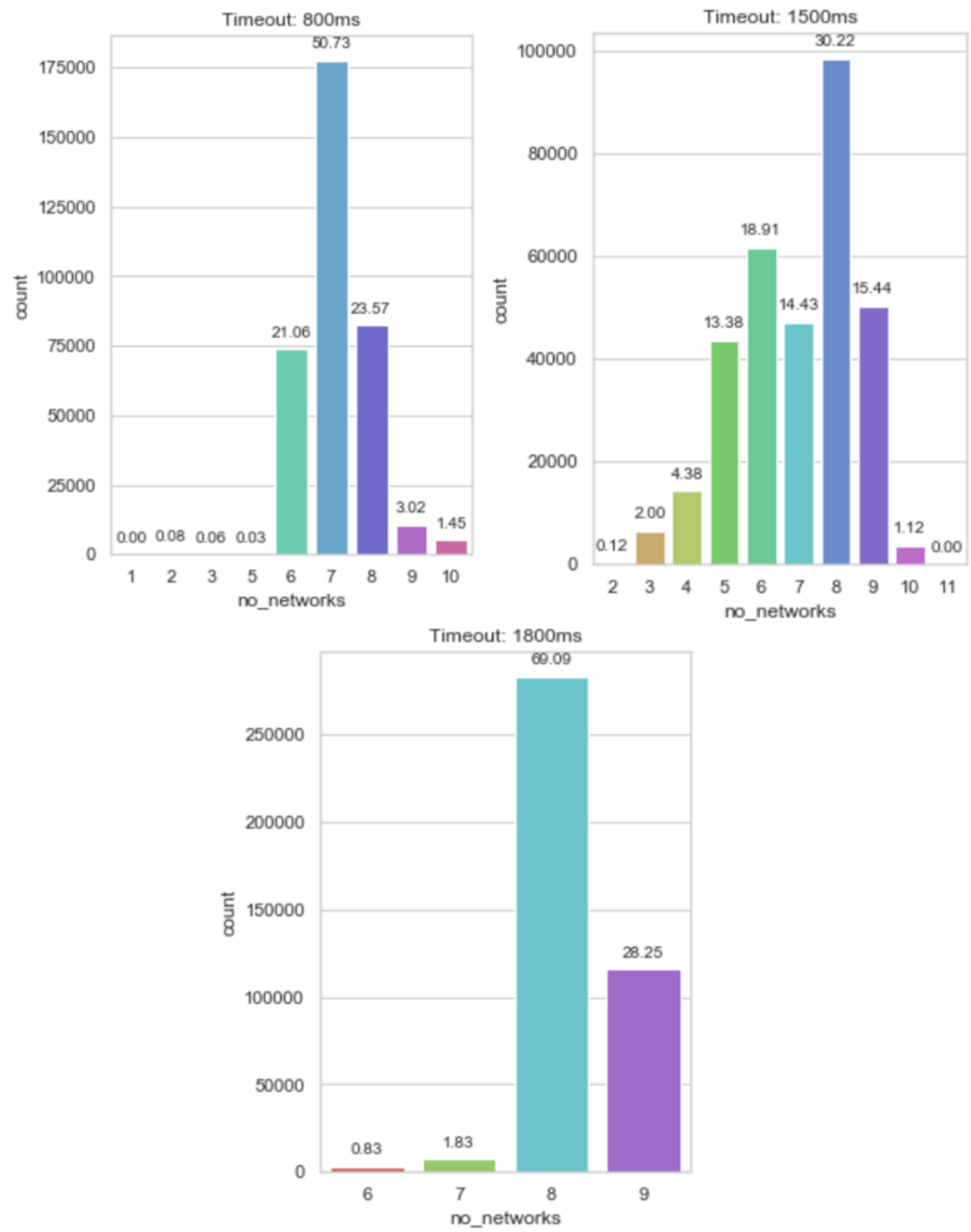

Figure 9: Number of networks distribution by percentage (%)

Timeout 1500ms set seems to have the widest variety of network numbers, ranging from 3 to 10 networks bidding for a request. Meanwhile, Timeout 800ms and timeout 1800ms sets mostly contain bids with more than 6 participating networks.

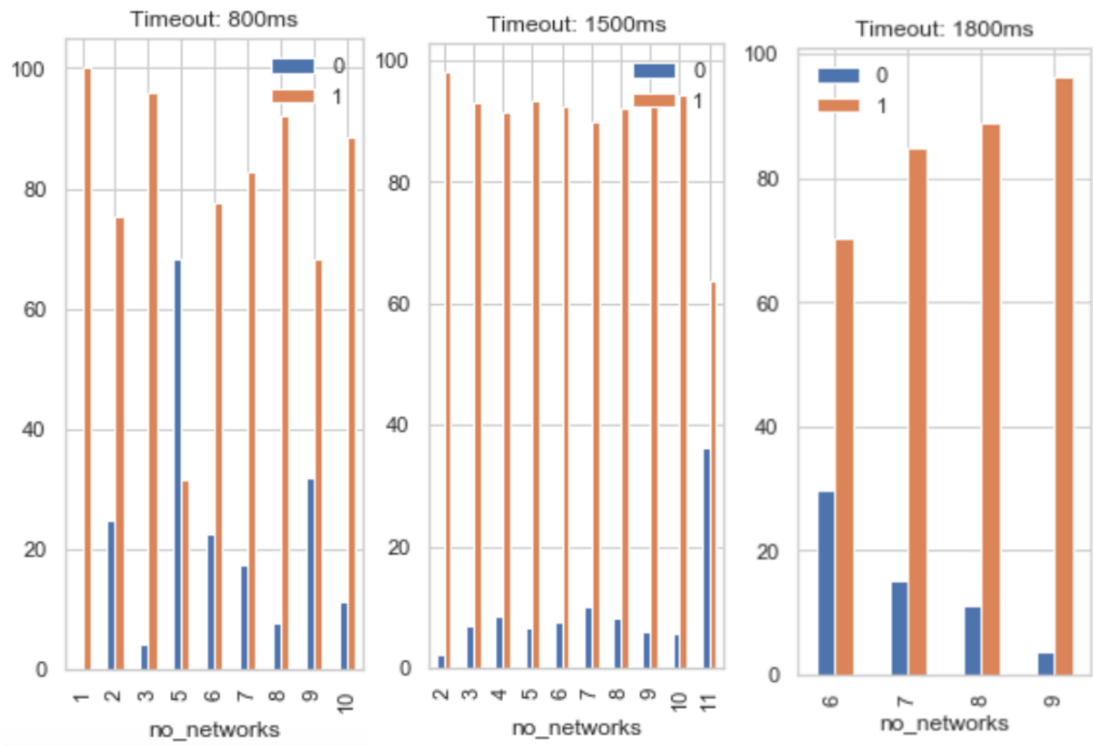

Figure 10: Time-out percentage by number of networks

From figure 10, we can see that the Timeout 1500ms set has very low timeout rates compared to the other 2 sets. Only the 11-network category has a high timeout rate but it is not an issue due to the small sample size of that category in the set, according to figure 9 above. In the Timeout 1800ms set, the timeout rate decreases when the number of networks increases, which is an interesting phenomenon since our literature suggests the opposite. The timeout rate in the other 2 sets moves more randomly. Visually, there is no common pattern of timeout rate that can be detected across all datasets. The timeout rate does not always increase or decrease when the number of networks increases.

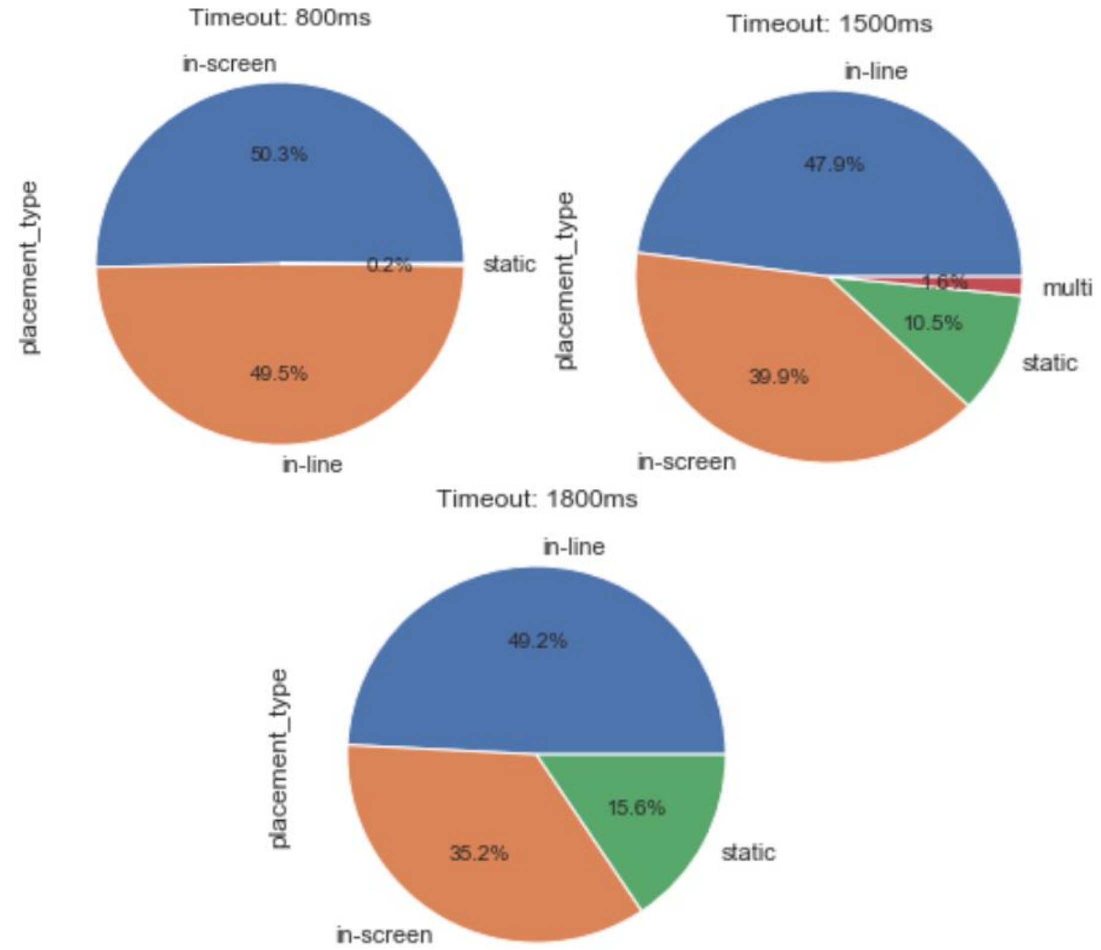

Figure 11: . Placement type distribution by percentage (%)

As can be seen, in-screen and in-line dominate all 3 datasets with 35-50% for each category. Even though there is a difference in order, sometimes in-screen is more popular and sometimes in-line has the majority but the difference between two categories is not substantial. Static is an important category in the last two datasets, occupying more than 10% of the total bids while it is unnoticeable in the first set, making up for only 0.2%. Multi is the category that only appears in the Timeout 1500ms dataset, due to its unpopularity in Kiosked's set-up. It is the newest placement type that only applies to a small group of clients.

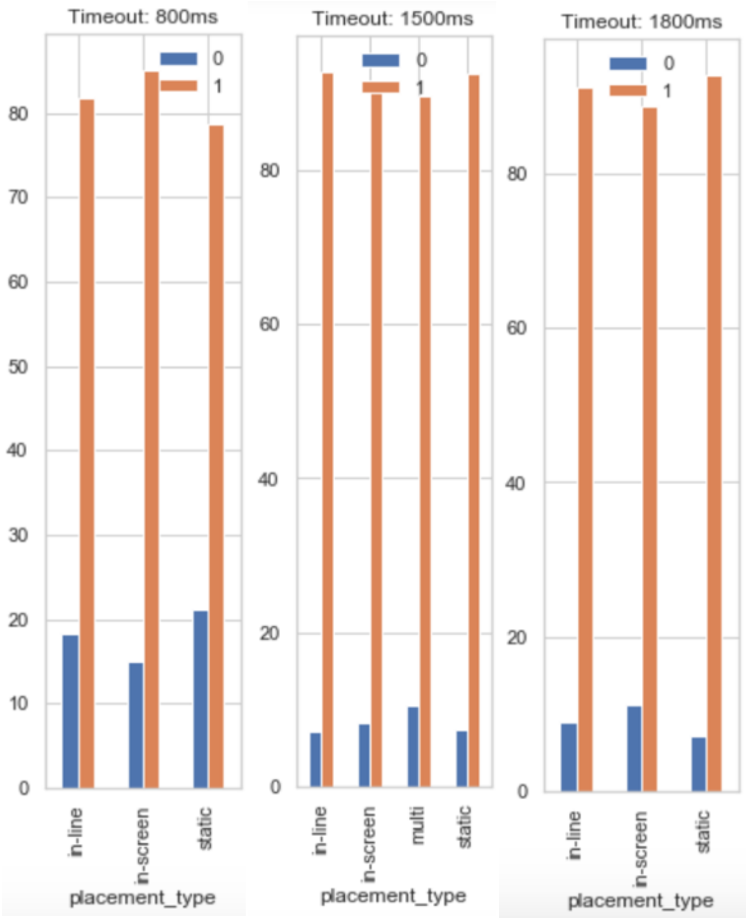

Figure 12: Time-out percentage by placement type

Figure 12 shows the same trend compared to the other Time Out percentage figure (Figure 5) that the first set has significantly higher timeout rates in all categories. However, each set has a very different ranking for the highest/lowest timeout rate. For example, the placement type with the highest timeout rate is static in the first set, multi in the second set and in-screen in the last set. In-screen placement has the lowest timeout rate in the first set but ranks high in timeout rate in the last 2 sets. In-line is the most stable placement type, always has middle ranking in all three datasets' timeout rate.

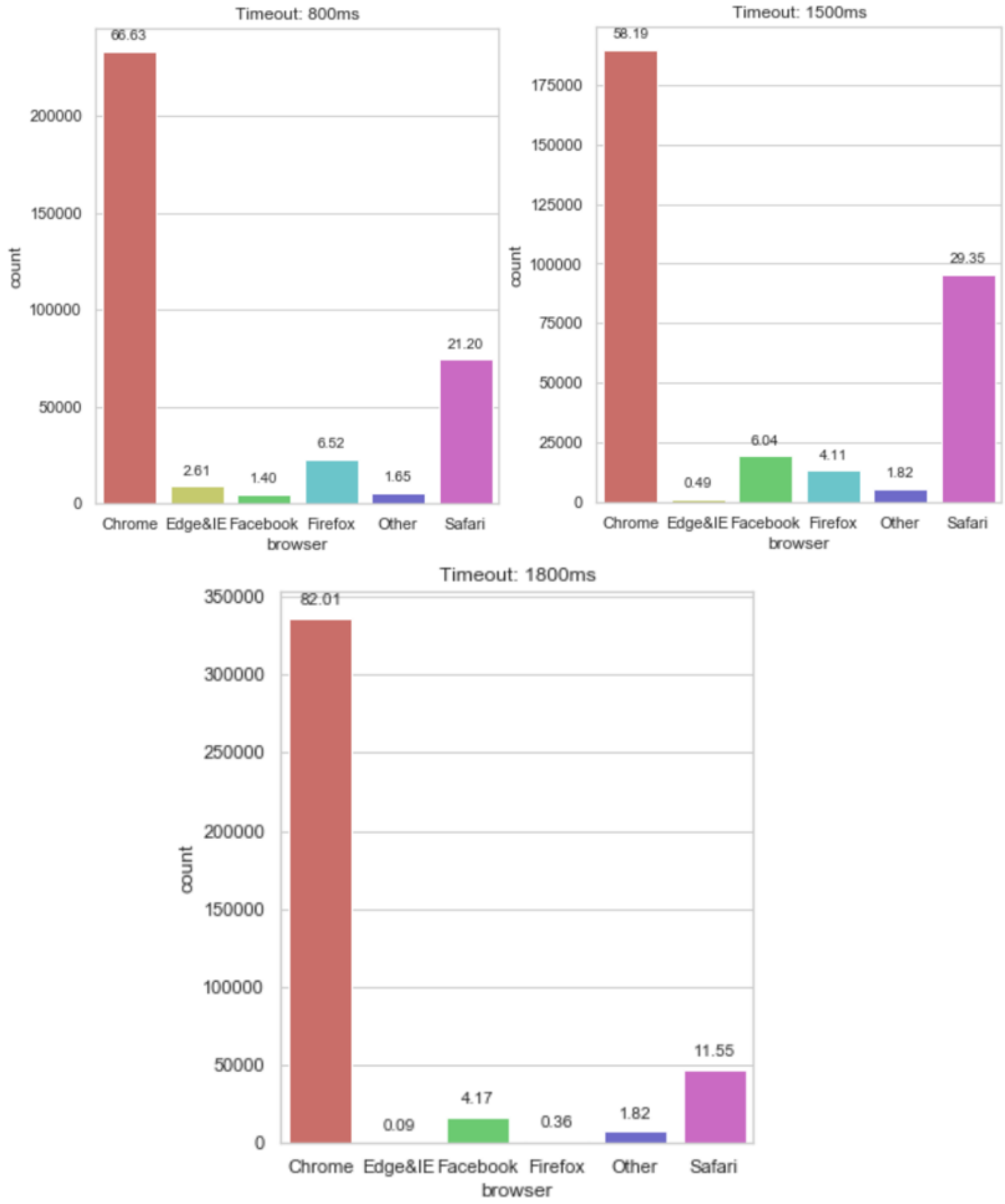

Figure 13: Browser distribution by percentage (%)

From figure 13, we can see that Chrome and Safari are the two most popular browsers across all datasets. Facebook is not popular but still present in all sets with 4-6% while Firefox's popularity decreases from first to second and third set, from 6.5% to almost none. Microsoft's browser group, Edge & IE, occupies a small percentage in all sets, highest in Timeout 800ms set with 2.61% and lower in the other two with <0.5%. The "Other" group is quite stable, making up for 1.5%-2% in each set.

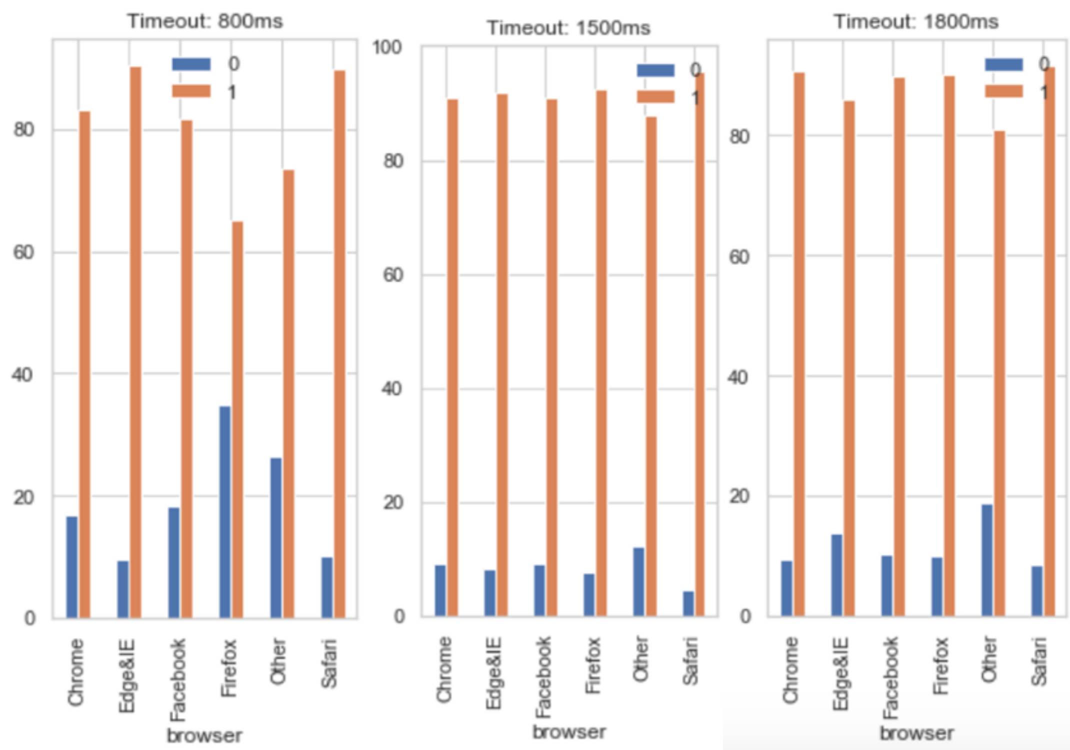

Figure 14: Time-out percentage by browser

Overall, the ‘Other’ group has very high timeout rates in all datasets, highest in the last two sets and second highest in the first set. It could be explained that ‘Other’ contains small minor browsers that are not as well-designed for javascript loading as major browsers. It could also result from the more attention given by the engineering team for major browsers when configuring the script compared to minor browsers so the scripts run more smoothly and with less error in major browsers. Between the two major browsers, Chrome and Safari, Safari has lower timeout rates in all datasets. The difference is small in the third set but very substantial in the others. Firefox’s timeout rates are also interesting to analyze. It is very high in the first set, almost double the set’s average rate of 16%, according to figure 5, but drops to similar or even lower than average in the last two sets. It seems that 800ms is too short for the bid requests on Firefox to get response. If it is possible to set separate timeouts per browser, it could be worthwhile to consider setting longer timeout values for Firefox browser.

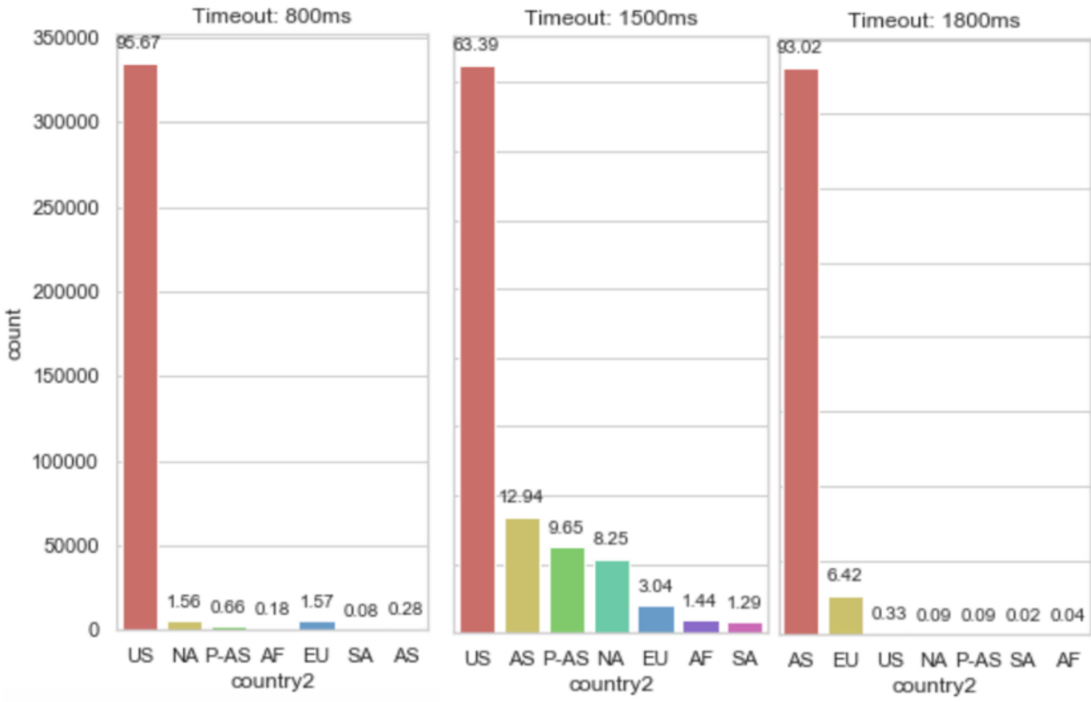

Figure 15: Geographic distribution by percentage (%)

It is clearly illustrated in figure 15 that the Timeout 800ms and Timeout 1800ms sets focus heavily on certain geographic regions, with the former containing mostly traffic from the US while the latter receiving the vast majority of bids from Asia and Europe. Timeout 1500ms is the relatively most balanced set with significant amount of traffic from several regions, even though there is still one dominant region (US) that occupies more than 60% of all traffic.

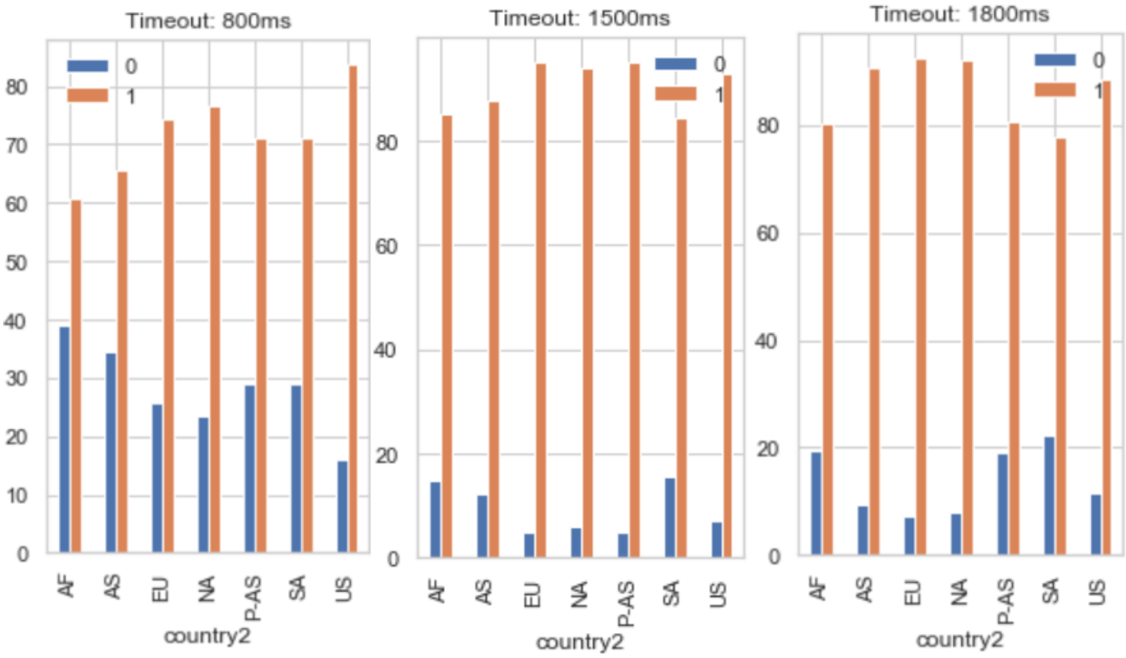

Figure 16: Time-out percentage by geographic regions

At the first glance, there is some timeout pattern by geographic region. Africa and South America always have high timeout rates compared to other regions in the same datasets. The US, North America and EU constantly appear in the group of low time-out-rate regions. Interestingly, Premium Asia group does not show significantly lower timeout rate compared to Asia group in all datasets as expected. Its timeout rate doubles Asian rate in the last dataset. However, it is hard to conclude anything from the graphs in this case as the sample size of each category differs greatly across datasets with many categories only occupying less than 1%, which is less than 500 samples.

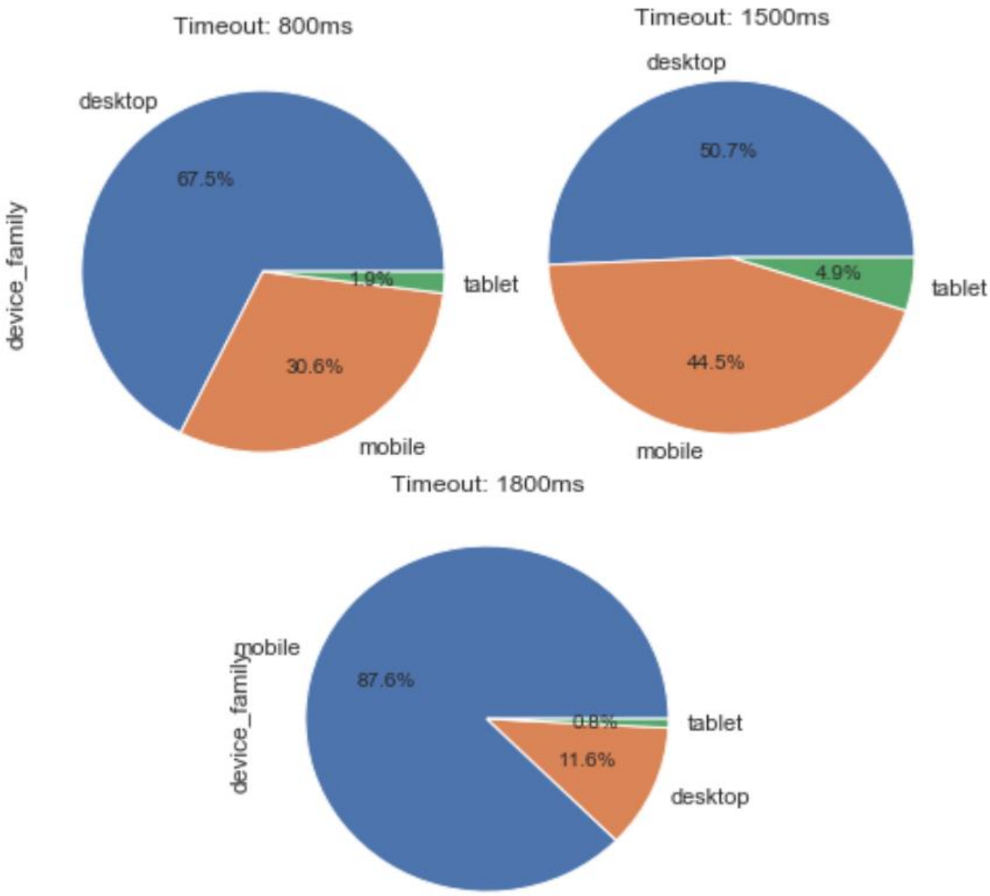

Figure 17: Device family distribution by percentage (%)

The first two datasets have the same order of device popularity. The desktop is the most popular device, occupying at least 50% of the set while mobile at second place is not so far, ranging from 30 to 45%. Even though the tablet is always the least popular device, it still occupies for a noticeable percentage of 2-5%. On the other hand, the 3rd dataset has a totally different distribution. Mobile occupies more than 87% of the set while desktop is far behind, only making up for 11%, and tablets almost disappear with less than 1% distribution.

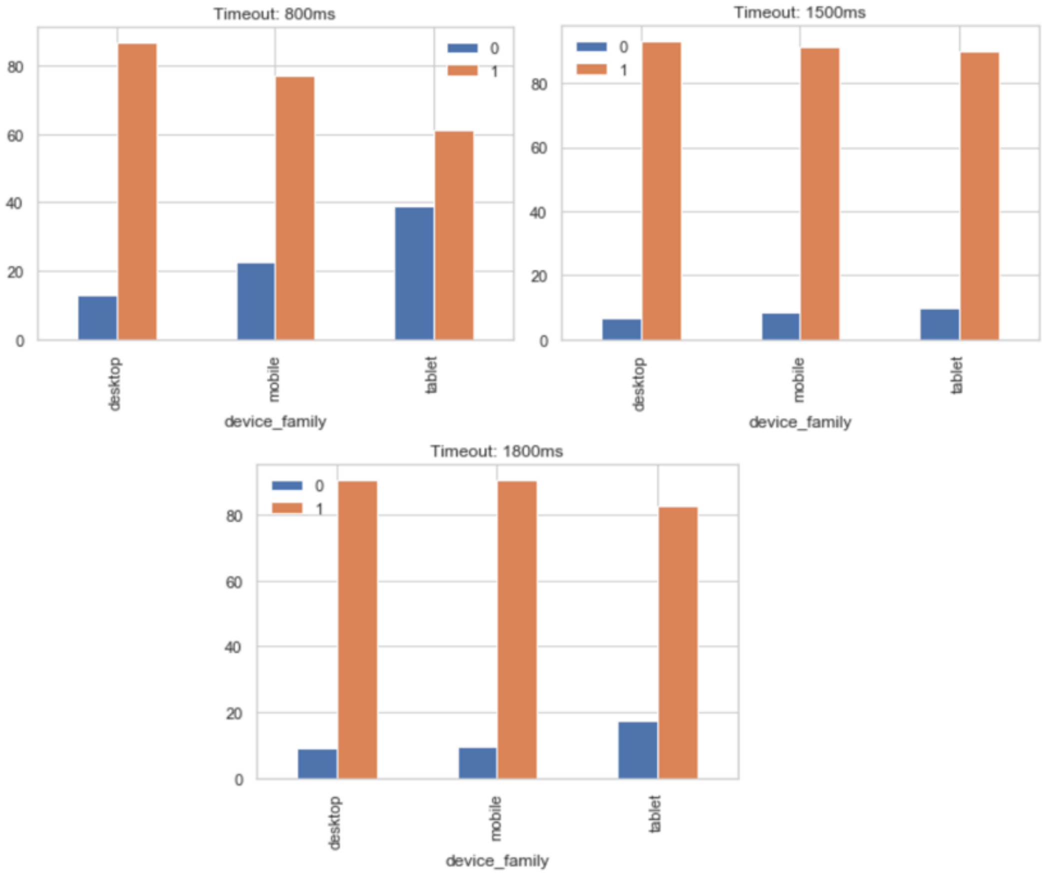

Figure 18: Time-out percentage by device family

Figure 18 shows that the Timeout 800ms set has a higher timeout rate in all device families compared to the other two sets. Timeout 1500ms and Timeout 1800ms do not have a big difference in overall timeout rate across devices. Tablet is the category with the biggest timeout rate in all 3 datasets, especially in the first set when its timeout rate is nearly 40%.

5 Predictive Models

5.1 Computational tools

The data is accessed by SQL WorkbenchJ, an application developed by Oracle. That is the most convenient way to access data in Kiosked's database and export them for further usage in other applications. According to Groff, et al. (2002), SQL is a powerful and easy-to-use tool that is free and can be installed quickly to personal computers. SQL uses simple commands that are close to normal English language; provides portability in accessing common databases and allows multiple views of the data (Groff, et al., 2002).

The data is then processed by Jupyter Notebook on Anaconda-Navigator platform. According to Braun, et al. (2017), the use of Jupyter Notebook is beneficial thanks to the high readability, ease of sharing and the ability to run cell by cell modification separately. The last advantage is very important with this thesis because of large datasets that the model needs to load and the long trial-and-error process that the author went through to build the model. If I need to rerun the full script to see the result of all minor modifications, it will be very time consuming.

Anaconda Navigator is the desktop graphical user interface (GUI) that makes it simple for Python users to manage and deploy packages and package versions without the use of command lines (Biswas & Datta, 2019). The interface is easy to install and use without the need for admin privileges.

5.2 Models

There are five different models that are considered for the Recursive Feature Elimination (RFE) process: logistic regression (lr), perceptron (per), decision tree (cart), random forest (rf) and gradient boosting (gbm).

Logistic regression is a supervised learning method that uses logistic function to measure the probability of relationship between explanatory variables and dependent variables. According to Sluijmers (2018), logistic regression is an easy to implement and understand method that does not require large computational capabilities. It is also easy to interpret the results thanks to the statistical summary table available in Python's statsmodels package. However, the model is prone to overfitting, having difficulty with the non-linear problem and unable to handle the large number of explanatory variables (ibid). In this thesis, we do not have an excessive amount of explanatory variables but we still need to test to see how logistic regression performs compared to other algorithms.

Perception is a simple classification algorithm which is similar to the logistic regression. The difference between the two algorithms is the loss function. According to Pedregosa, et al. (2011), perception is suitable for learning on a large amount of data and it is quite fast to train.

According to Pedregosa, et al. (2011), decision tree is a supervised learning method that presents the data features into a tree-formed model with a series of true/false decision rules. Decision tree is one of the most popular machine learning algorithms thanks to its simplicity to implement, interpret and visualize (ibid). The method requires very little data pre-processing and provides statistical tests for validation (ibid). However, the decision tree method also has some disadvantages. The method could result in an over-fitting model, a model that fits very well with the training set but predicts very poorly on the test set. In addition, it is also unstable, cannot work with missing values and easily biased if the dataset is not balanced (ibid). The last disadvantage is solved in this thesis's dataset because we already balanced the data by SMOTE prior to fitting the model. The overfitting can be solved by using random forest.

Random forest is similar to the decision tree method but in the random forest method, many trees are trained from random subsets of the training set instead of just one tree and the outcome is the average result (ibid). The randomness in choosing subset and the average outcomes reduce variance and overfitting errors, solving most of the problem of the decision tree. However, by that increasing complication, the simplicity and ease to interpret the decision tree method do not stay. That is the disadvantage of random forest compared to decision trees. Besides, even though the feature importance ranking of random forest in the sklearn package is very useful for this thesis' purpose, it is supposed to favor the feature with more unique values (ibid), which is the number of networks.

Gradient boosting is another tree-based algorithm. Instead of training independent trees like random forest, the tree in gbm is trained one by one with the next one improving the deviance in the previous one (ibid). As a result, GBM is more time-consuming to build and more prone to overfitting. On a positive side, GBM has a good support for missing value and the sequential building of trees could help to reduce the error rate (ibid).

5.3 Implementation

According to the descriptive analysis, we can see that Timeout 1500ms is the most balanced set that has all representatives from all categories with a relatively substantial percentage. The other two sets have some categories that strongly focus on certain values and almost or totally omit other values. As a result, we choose to build the model on the Timeout 1500ms dataset to include as many variables into consideration as possible with large enough sample size for each variable so that they can be statistically significant.

After creating dummy variables and removing one dummy variable for each category to avoid the dummy variable trap, we are left with 27 columns - 1 dependent variable and 26 explanatory variables. However, 26 explanatory variables are quite a large amount and could cause overfitting for the model. Lever, et al. (2016) discusses overfitting and underfitting issues in logistic regression models. According to the article, if there are too few variables included in the model, the model could be too simple and highly biased. On the other hand, if there are too many variables included, the model could become too complicated, learn from noise, irrelevant variables, and require more computer's processing power to process unnecessary data (ibid). As a result, it is important to do feature selection to check the available variables and choose only relevant ones. The thesis utilizes the available RFE (Recursive Feature Elimination) algorithm in sklearn to test and evaluate the variables.

6 Results

6.1 Model Evaluation & Feature Elimination

First, to determine how many variables should be included in the model, we run the loop for the amount of variables from 10 to 27. The temporary choice of estimator is the Decision Tree algorithm (CART) since it does not require normalization, has no assumption of the data distribution and works well with already-balanced data (Navlani, 2018).

```
def get_models():
    models = dict()
    for i in range(10, 27):
        rfe = RFE(estimator=DecisionTreeClassifier(), n_features_to_select=i)
        model = DecisionTreeClassifier()
        models[str(i)] = Pipeline(steps=[('s',rfe),('m',model)])
    return models
```

Figure 19: Get model set for different numbers of explanatory variables included

The result models are evaluated by K-fold cross validation. The number of splits is commonly set at 5 or 10 (Lever, et al., 2016). In this case, it is set at 10. The criteria used to evaluate the model is precision. The reason why precision is used instead of the common accuracy criterion is the thesis' focus to explore timeout bids instead of finished bids. Precision is calculated as: Precision = True Positives / (True Positives + False Negatives). As a result, the higher precision is, the lower False Negatives rate is. By prioritizing on precision, we will have a model with best power to identify timeout bids (negative outcomes).

```
def evaluate_model(model, X_train, y_train):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X_train, y_train, scoring='precision', cv=cv, n_jobs=-1, error_score='raise')
    return scores
```

Figure 20: Evaluate models to choose optimal amount of variables

The result is illustrated in the box plot as follow:

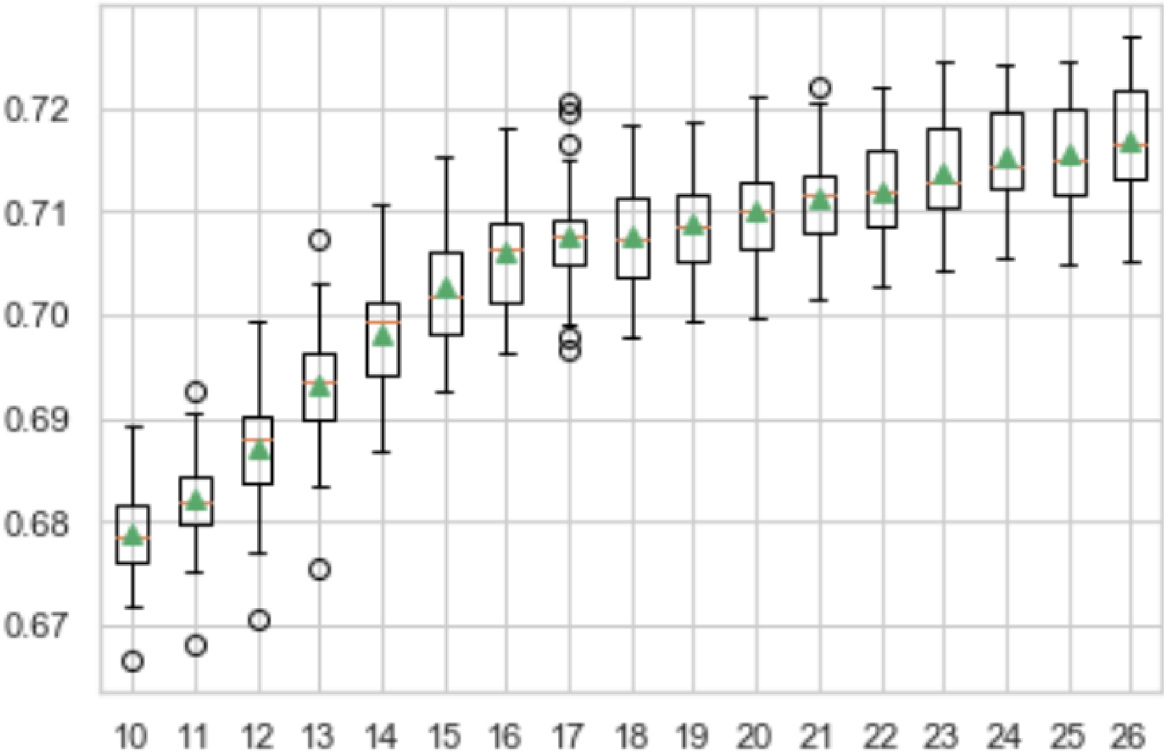

Figure 21: Boxplot of models' precision with different amounts of variables included

As can be seen from figure 21, 26 is the amount of variables that make the most precise model. However, the difference in mean precision between model with 23 variables and model with 26 variables is quite small. So we could use 23 variables only in the final model to reduce complexity and increase training speed.

The next question is which estimator should be used. RFE can be run with a wide range of estimator algorithms with different pros and cons. To determine which algorithm gives the highest precision model, we run another test to compare between 5 common algorithms for classification models: logistic regression (lr), perceptron (per), decision tree (cart), random forest (fr) and gradient boosting (gbm). All evaluated models try to find 23 most relevant variables from the 26 available ones. The models are also evaluated by k-fold cross validation with 10 folds setting and precision scoring just as in the number-of-variable evaluation above. The result is demonstrated in the boxplot below:

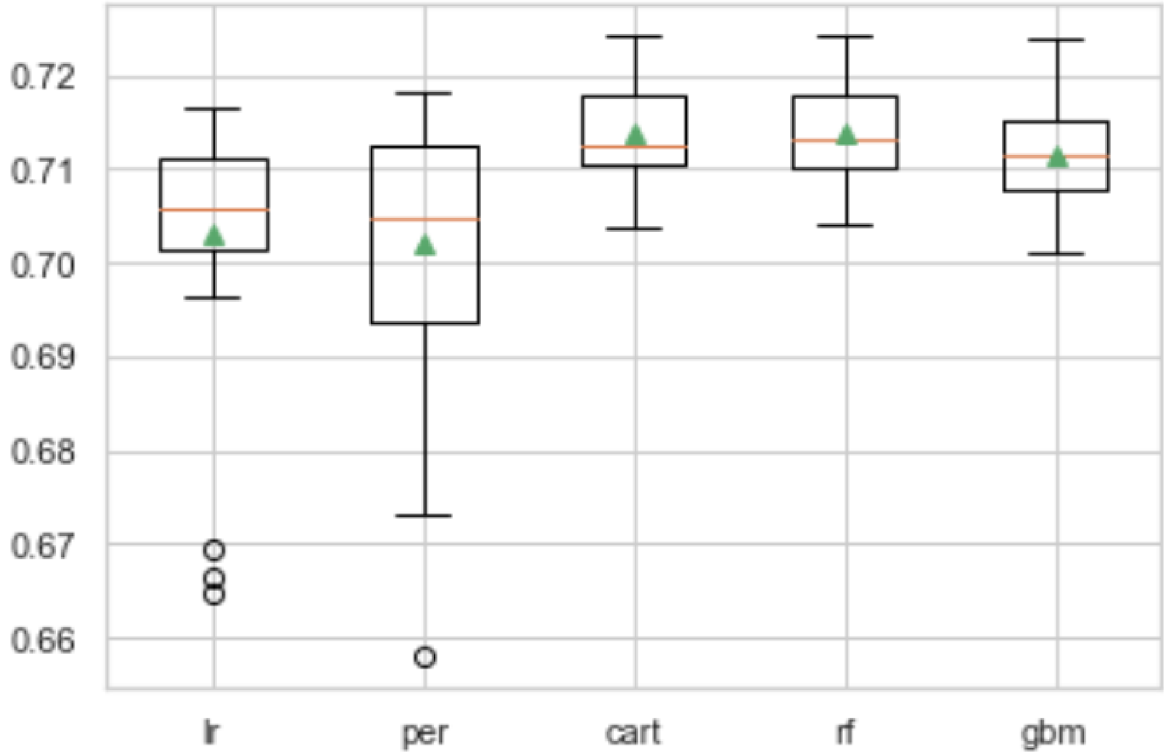

Figure 22: Boxplot of models' precision with different estimator algorithms

According to figure 22, models with decision tree and random forest estimators have exactly the same precision (mean and standard deviation) and also the highest precision among all 5 models. Models with logistic regression and perception are clearly worse with lower mean precisions and wider standard deviation range while the model with gradient boosting has slightly lower mean precision compared to the best ones and exactly the same standard deviation. There could be doubt that the Decision Tree performs best in this test because it is used to select the amount of variables so this amount of variables fit the algorithm well. To avoid that bias, random forest is chosen as the estimator for feature selection in the final model.

Figure 23 below shows the feature ranking by random forest. The higher importance value a feature has, the more relevant it is to the dependent variable. The figure includes ranking information of all available features and the four last features: placement_type_multi, browser_Edge&IE and bid_network_smartadserver will be excluded from further models.

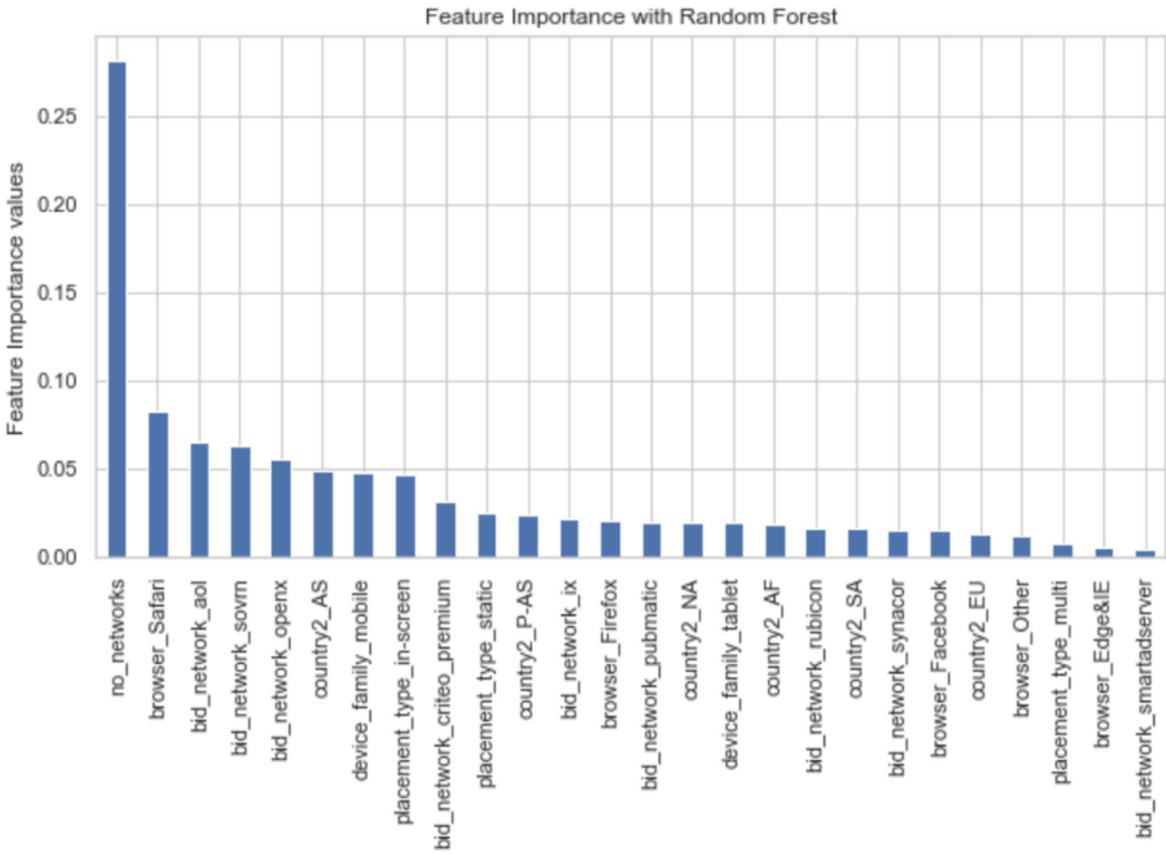

Figure 23: Explanatory variables ranked by feature importance values

In the final step, the model is built with logistic regression in statsmodels package as in this step, we also want to know each variable's coefficient and statistical significance and that is what statsmodels is better at compared to scikit-learn (Boland, 2017). In previous steps, we focus on finding which models give the best performance to choose the most relevant subsets of features while in the final models, we also concentrate on how each feature relates to the bid_result and whether those relationships are statistically significant.

6.2 Testing for significance

According to table 2, not all variables included have statistically significant relationships with the bid_result. We choose the p-value threshold of 0.05 or confidence level of 95%. Four variables: browser_Other, browser_Firefox, placement_type_in-screen and bid_network_rubicon have much larger p-values compared to the threshold. In those cases, we cannot reject the null hypothesis that there are no relationship between the variable and the bid results. As a result, those four variables are excluded from the future discussion.

Table 2: Summary of the Logit model results

Variable	**Coefficient**	**P-value**	**Standard error**
no_networks	0.0523	0.0000	1.0827
browser_Facebook	0.3621	0.0000	0.0315
browser_Firefox	0.0154	**0.6949**	0.0391
browser_Other	0.0198	**0.7084**	0.0529
browser_Safari	0.7546	0.0000	0.0195
device_family_mobile	-0.3728	0.0000	0.0162
device_family_tablet	-0.7387	0.0000	0.0352
placement_type_in-screen	0.0240	**0.2085**	0.0191
placement_type_static	0.3705	0.0000	0.0253
country2_AF	-0.5833	0.0000	0.0570
country2_AS	-0.5046	0.0000	0.0208
country2_EU	0.5625	0.0000	0.0524
country2_NA	0.2563	0.0000	0.0288
country2_P-AS	0.3844	0.0000	0.0294
country2_SA	-0.7224	0.0000	0.0597
bid_network_aol	-0.8373	0.0000	0.0231
bid_network_criteo_premium	0.3018	0.0000	0.0277
bid_network_ix	0.2088	0.0000	0.0290
bid_network_openx	-0.9655	0.0000	0.0271
bid_network_pubmatic	0.4792	0.0000	0.0383
bid_network_rubicon	-0.0237	**0.3845**	0.0273
bid_network_sovrn	-1.0759	0.0000	0.0281
bid_network_synacor	-1.0827	0.0000	0.0542

Other variables are relevant and can be interpreted by the coefficients in reference to the excluded variables: browser_Chrome, device_family_desktop, placement_type_in-line, country2_US and bid_network_appnexus. As all variables except for the number of networks are binary so their strength of influence on the bid results will be interpreted from the coefficients. The importance of the number of networks factor will take into account the value of the column as well as the coefficient.

According to the summary table, the number of networks has positive effects on the bid's completion: the higher the number of participating networks, the less likely that the bids get timeout. Even though the coefficient is not high, the effect of this variable is still quite strong since it multiplies with a large amount of networks (mostly ranging from 5 to 9 networks in the dataset).

In the browser group, Safari and Facebook are proved to have better effect on the timeout status than Chrome, especially Safari. Bids requested from Safari browser have substantially higher chances to receive answers from networks. This result confirms the observation from Figure 14. that Safari has a constantly lower timeout rate compared to other browsers.

The opposite trend is shown in the device family group's variables. Both mobile and tablet show negative influences on the timeout rate compared to desktop with mobile devices moderately increasing the chance of bid timeout and tablet devices significantly boost that chance. While the tablet's result confirms the observation from Figure 18, mobile's result indicates the better bid completion on desktop even though that pattern is harder to detect from the graph.

The difference in effect of placement types on bid results is not as noticeable as in the two groups above. Both in-screen and static placement types have slightly positive effects on the bids' timeout status. That difference could be the result of the way each type of placement is run or shown on the screen. As static placement is run through publishers' frames instead of through third party's (Kiosked's) frames, it could be more compatible with the sites' configuration and be lighter to run. On the other hand, both in-screen and in-line are served through Kiosked's frame but in-screen is loaded on the top frame of the site while in-line is loaded under certain elements, especially rich media. As a result, the element loading process could take some bandwidth and increase in-line bids' latency.

The country group indicates no surprise since the group of more developed countries like North America, Europe and Premium Asia show positive effect on bids' completion while the group of developing countries like Asia, South America and Africa shows higher tendency to go timeout when compared with the reference region (US). As mentioned in the data preparation part, this difference is expected due to the difference in infrastructure development level which could result in better connection speed.

In the last group, except for rubicon which is not statistically significant, other networks are divided into two groups. The group that are less likely to timeout compared to appnexus includes criteo premium, ix and pubmatic while the group that tends to timeout more consists of aol, open, sovrn and synacor. This result again fits perfectly with the observation in Figure 8.

6.3 Predictive test

After fitting the model, we come to another important step of utilizing it to predict timeout status in the test set. The prediction is compared with the true y_test value and the result confusion matrix is as follow:

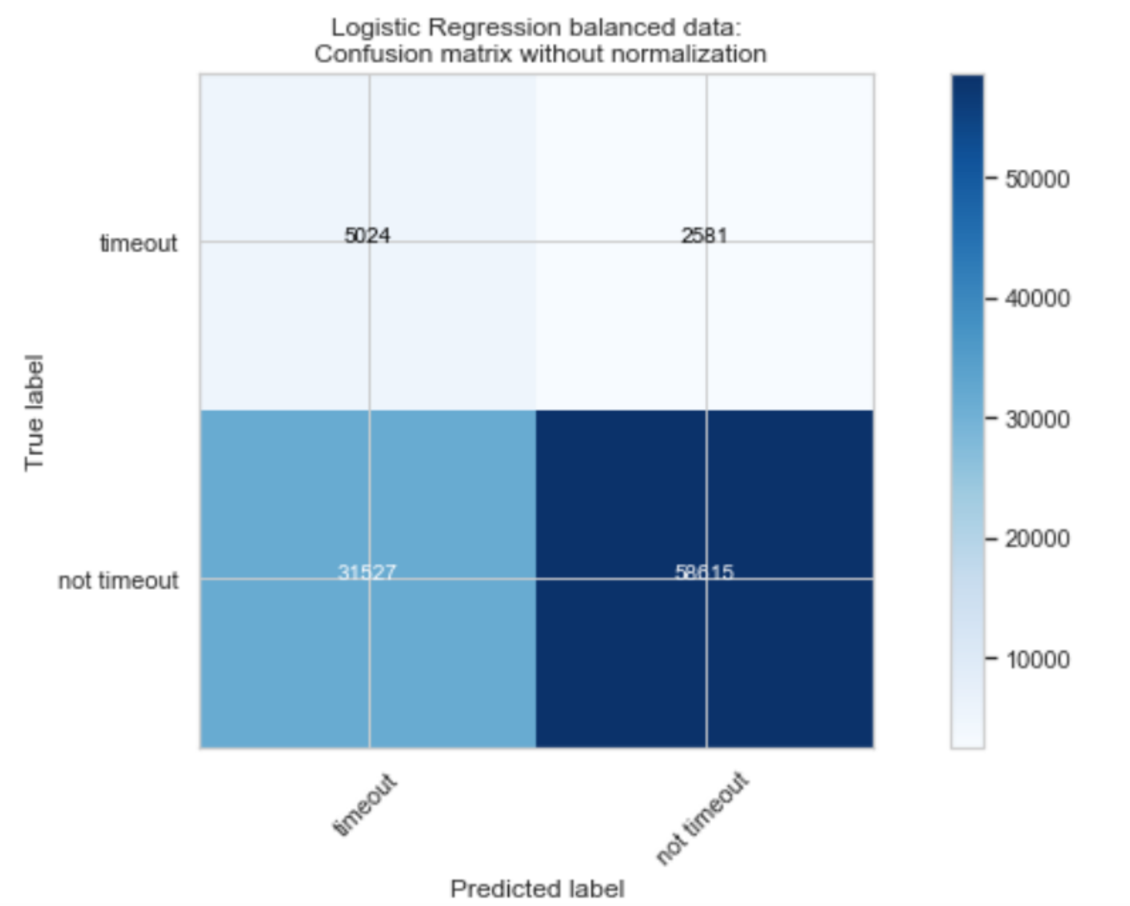

Figure 24: Confusion matrix of the predictive model

Figure 24 shows the amount of bid result predicted as timeout and not timeout (Predicted label) versus the real result of the bids (True label). According to the result, the model predicts correctly approximately twice as much as incorrectly. It is good to notice that the imbalance distribution between timeout and not-timeout result in the train set does not have significant negative effect on the prediction (the prediction is not entirely not-timeout), thanks to rebalancing step and choosing precision as the scoring evaluation.

Table 3: Classification report on the predictive model

	precision	recall	f1-score	support
0	0.14	0.66	0.23	7605
1	0.96	0.65	0.77	90142
accuracy			0.65	97747
macro avg	0.55	0.66	0.50	97747
weighted avg	0.89	0.65	0.73	97747

Table 3 shows that the accuracy of the model is 65% which is not excellent but we have quite good precision when predicting around 66% of the timeout cases correctly (recall = 66% for 0-value). As bid_result_timeout is the minor outcome that we focus on, the model is considered to deliver satisfactory predicting power.

As the model has satisfactory predicting power, we could use it to weigh the likelihood that a bid can timeout based on relevant variables and adjust the timeout limit accordingly to maximize performance.

7 Discussion & Conclusion

7.1 Revisiting research questions

The first objective is to find out which factors affect the bids' result. Both the descriptive analysis and the predictive test suggest that all groups have some relationship to the bid result. According to the descriptive analysis, no group has the similar timeout rate among all variables, even though the timeout rate ranking of the variables in the same group differs between 3 datasets. On the other hand, the model summary (Table 2.) suggest that all groups have at least one variable with statistically significant impacts on the bid results but not all variables. The groups with all variables affected bid result are number of networks, device family, placement type and country. The groups with only a part of the variables affecting bid results are browser (only Safari and Facebook) and bid network (all except for rubicon and smartadserver).

With regards to the strength of the effects, both the random forest feature ranking (Figure 23) and the model summary (Table 2) agree that number of networks, browser

Safari, bid network aol, sovrn and openx are highly important. The ranking of the number of networks factor by random forest is high as expected as the random forest ranking is known to favor variables with more unique values. All other variables are binary so the number of networks is the only one with more than two unique values. However, when considering the coefficient in the logistic regression report, the number of networks still has a significant effect on the bid result. The random forest ranking also suggests that mobile devices, Asia geo region (AS) and placement type in-screen have strong relationship with the bid result. The logistic regression model rejects the placement type in-screen as not statistically significant while placing Asian region and mobile device into the group of factors with moderate impacts. The logistic regression model suggests that tablet device, South America (SA) and bid network synacor are highly relevant to the bid result while the random forest RFE ranks them much lower, mostly due to their small sample size in the dataset.

When it comes to the direction of the effects that those factors have on the bids' results, most factors react as expected from the literature and the descriptive analysis. The factors that have relatively positive effect on the bid result are the number of network; Safari and Facebook browser; static placement type; premium-Asia, EU and North America regions; criteo premium, pubmatic and index exchange bid networks. On the other hand, the factors that have relatively negative effect on the bid result include tablet and mobile devices; Asia, Africa and South America regions; aol, openx, synacor and sovrn bid networks.

Number of networks factor is the only one that shows unexpected results, indicating that the increasing number of participants reduces the timeout chance. It is against the literature's suggestion that the bigger number of participating networks will increase the bid latency and result in higher timeout rate. The reason for the positive effect may rely on some non-quantitative factors that are not included in this test and could influence both the number of networks and the bid_result. Sites' quality is an example. A high quality site is more likely to receive approval from networks, making the number of bidding networks increase while bid timeout rate could be reduced on that type of sites thanks to better structure design and/or optimal technical configuration. In addition, the test in the literature review includes a large number of sites (50%) with only 1 demand partner and compared that to other groups with more than 5 and more than 10 partners. The wide range in the number of partners may show the negative effect more clearly. Meanwhile, in our case, the number of networks mainly range from 5 to 10 networks, which is a comparatively smaller difference. The smaller difference may diminish the negative impact (if any).

7.2 Managerial implications

From the findings above, we can conclude that some set up features have significant impacts on the bid results. Even though we cannot remove the bids that relate to factors that have negative effects, we can set longer timeout values for them. On the other hand, when a script has most traffic from sources with positive factors such as Safari browser, EU region, desktop device and placement type static, we can set a shorter timeout value to speed up the auction. To make it possible, the company should consider allowing different timeout values to be set up per browser, geographical region and/or devices instead of only per script as for now.

In addition, there seems to be a substantial difference in speed among bid networks. Some networks, such as aol, sovrn, synacor or openx, are much slower than others. Bid networks are trickier than other factors because setting timeout for each network may not improve auction timeout since the auction needs to wait for all networks to either answer or timeout to start. To solve the discrepancy in timeout for different networks, it is recommended to feedback to and work with the slow networks to find out if there are any issues in configuration which could slow down the bidding process.

7.3 Limitations of the study

The thesis has several limitations due to the computational capacity, data availability and the choice/assumption made when selecting the data..

First, due to computational capacity, the amount of data processed is modest compared to the amount of data available. Even though 326,133 bids are analyzed, it is still a very small number compared to the 2.5 billions bids available. The dataset is also resampled down to around 50,000 bids to have balanced representation of timeout and not timeout bids. Not to mention, that 2.5 billions bids only reflect 6-month data of the company. So the analysis is likely to omit seasonal differences (if any) and some variables are not closely evaluated. For example, bids through tablet devices, vertical size or multi placement type are rare in the sample set, making it difficult to conclude about their effect on the bid result and determine an optimal timeout value range for them if necessary.

Second, as mentioned in the literature review part, real-time bidding in general and timeout performance in particular are pretty new topics so there are not many academic papers to use as reference. As a result, there may be potential factors that are not considered in the analysis. For example, the placement type factor is included without academic background thanks to the practical observation but there are likely other relevant factors that could go unnoticed due to the lack of academic reference.

Third, due to the availability of the data that Kiosked collected, some potential factors are not considered. For example, the number of requests per load (Pachilakis, et al., 2019) and the hour time of the connection (Bauer, et al., 2010) are mentioned as potential determinators of network connection speed. Unfortunately, there are no records of the number of requests per load in Kiosked's system. The hour time is recorded but it is quite tricky. The time recorded in Kiosked’s system is already converted to UTC time, which is meaningless in determining the situation of the users’ network connection (peak time or off-peak time). It is also not possible to convert the time back to local time due to the lack of exact location information. Big countries like the US, Australia or Canada all have many different time zones so with only UTC time and the country information, it is not possible to get reliable local time.

Fourth, the country groups are based on common continent classification which implies an assumption that timeout in the same continent group is similar. Only the Asia group is divided into two separate groups because the difference in performance between those countries’ websites and the rest of Asian websites are so easily observable. Even though the model shows that the country groups have statistically significant influence on the bid result, there could be similar discrepancies within other groups that are not yet acknowledged. If the country groups are categorized in more details, the differences between groups could be clearer and they could have higher values in the predictive model.

Finally, to avoid the differences caused by different timeout values, the dataset only focused on the bids with timeout values of 1500ms. Even though the Timeout 1500ms set is the most balanced set, the choice of using that set only could underestimate factors that are more popular in other sets. For example, traffic from Asia and EU regions are more popular in the 1800ms set and the 800ms set has more bids of in-screen placement type.

7.4 Suggestions for future research

As stated earlier, the real time bidding topic with the focus on timeout has not yet been thoroughly discussed by academics so there are a wide range of opportunities remaining for future research.

First, it could be really beneficial to look closely into the geographical regions and divide them into categories with more homogenous countries. It could be also interesting to look into more detailed levels than countries, like cities or regions because the network connection quality within a country is not always the same.

Second, if the data system allows collecting amounts of ads per page and the local time of the connections, it could be fascinating to see how those factors affect the bid results. Besides, the dataset in this thesis has some unpopular factors such as tablet devices, South America/Africa regions or Edge/IE browsers which may have clearer impacts on the bid result if their sample size is increased. As a result, some further research could be carried out with the focus on those factors, which could be beneficial for ad tech companies with more traffic from those mentioned regions.

Last but not least, this thesis found out the contradictory effect of the number of networks compared to previous studies. It could be interesting for further studies to look into this phenomenon to find out the reason. There are two ideas that could be considered. One is to test on a wider range of network numbers, for example testing on bids with 1 to 15 networks instead of 5 to 10 networks as in this paper. The other idea is to follow the timeout situation of the same website with a different number of networks. For instance, reducing the amount of bidding networks from 10 to 5 to see if bid result improves (less timeout) or if bid duration decreases significantly.

7.5 Conclusion

Real time bidding has become an increasingly popular option for online marketing in the last decade. There are some academic papers about the field but mostly focus on pricing model rather than technical aspects like timeout. This paper tested the hypotheses suggested by available academic research about factors that could affect the bid timeout result. By running logistic regression on the dataset provided by Kiosked, an ad-tech company, we found out that there are some connections between those tested factors and the bid results. Bids with certain characteristics get timeout more often than with other characteristics. It leads to the concern that the company should make it technically possible to set timeout in more detailed levels such as browsers, devices or countries because those factors influence the chance that the bids get timeout.

References

Bauer, S., Clark, D.D. and Lehr, W. (2010). *Understanding broadband speed measurements.* Tprc. Available from: https://www.researchgate.net/publication/228280854_Understanding_Broadband_Speed_Measurements [Accessed on November 11th 2020]

Biswas, R. and Datta, A., (2019) *Detection of Hate Speech on Social Media Using Machine Learning.* Available from:

http://dspace.daffodilvarsity.edu.bd:8080/bitstream/handle/123456789/4194/P15379%20%2828_%29_.pdf?sequence=1&isAllowed=y [Accessed on November 1st 2020]

Boland, S. (2017) *Scikit-learn vs. StatsModels: Which, why, and how?* The Data Incubator Available from: https://blog.thedataincubator.com/2017/11/scikit-learn-vs-statsmodels/ [Accessed on November 2nd 2020]

Braun, N., Hauth, T., Pulvermacher, C. and Ritter, M., (2017) *An interactive and comprehensive working environment for high-energy physics software with python and jupyter notebooks.* In Journal of Physics: Conference Series (Vol. 898, No. 7, p. 072020). Available from: https://iopscience.iop.org/article/10.1088/1742-6596/898/7/072020/pdf [Accessed on November 1st 2020]

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) *SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research*, 16, pp.321-357. Available from: https://www.jair.org/index.php/jair/article/view/10302/24590 [Accessed on November 1st 2020]
Dewitt, G. (2019) *Adaptive Timeout™ – Incorporating Adaptive Machine Learning into Header Bidding.* Index Exchange. Available from:

https://www.indexexchange.com/adaptive-timeout-incorporating-adaptive-machine-learning-into-header-bidding/ [Accessed on November 7th 2020]

Evans, D.S. (2009) The online advertising industry: Economics, evolution, and privacy. Journal of economic perspectives, 23(3), pp.37-60. Available from: https://pubs.aeaweb.org/doi/pdfplus/10.1257/jep.23.3.37 [Accessed on November 22nd 2020]

Groff, J.R., Weinberg, P.N. and Oppel, A.J., (2002) *SQL: the complete reference (Vol. 2).* McGraw-Hill/Osborne, 1(1), pp.11-12. Available from: http://englishonlineclub.com/pdf/SQL%20-%20The%20Complete%20Reference%20[EnglishOnlineClub.com].pdf [Accessed on October 31st 2020]

Kumar, J. (2017) *Timeout analysis, troubleshooting and notification in real time bidding advertising system with implementation*. Comput. Sci. Eng, 7(3), pp.67-78.

Lever, J., Krzywinski, M. and Altman, N. (2016) *Points of significance: model selection and overfitting.* Available from: https://www.nature.com/articles/nmeth.3968.pdf?origin=ppub [Accessed on November 1st 2020]

Navlani, A. (2018) *Decision Tree Classification in Python.* Data Camp. Available from: https://www.datacamp.com/community/tutorials/decision-tree-classification-python

[Accessed on November 1st 2020]

Nielson, J., Williamson, C. and Arlitt, M., (2008) *Benchmarking modern web browsers.* In 2nd IEEE Workshop on Hot Topics in Web Systems and Technologies. Available from: https://pages.cpsc.ucalgary.ca/~carey/papers/2008/Jordan-HotWeb2008.pdf [Accessed on October 21st 2020]

Pachilakis, M., Papadopoulos, P., Markatos, E.P. and Kourtellis, N., (2019) *No More Chasing Waterfalls: A Measurement Study of the Header Bidding Ad-Ecosystem.* In Proceedings of the Internet Measurement Conference (pp. 280-293). Available from: https://arxiv.org/pdf/1907.12649.pdf [Accessed on February 27th 2020]

European Union. (2016) L119, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, 59(1-88), p.3.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. (2011) *Scikit-learn: Machine learning in Python.* The Journal of Machine Learning Research, 12, pp.2825-2830.

Sayedi, A., (2018) *Real-time bidding in online display advertising.* Marketing Science, 37(4), pp.553-568. https://faculty.washington.edu/aminsa/papers/RTB.pdf [Accessed on February 27th 2020]

SelfAdvertiser (2017) *Ad Network Vs Ad Exchange.* Available from: https://selfadvertiser.com/blog/ad-network-vs-ad-exchange/ [Accessed on November 4th 2020]

Shen, J., Orten, B., Geyik, S.C., Liu, D., Shariat, S., Bian, F. and Dasdan, A., (2015) *From 0.5 million to 2.5 million: Efficiently scaling up real-time bidding.* In 2015 IEEE International Conference on Data Mining (pp. 973-978). IEEE. Available from: http://wnzhang.net/share/rtb-papers/turn-throatling.pdf [Accessed on February 27th 2020]

Sluijmers, M (2018) *Python (Scikit-Learn): Logistic Regression Classification.* Available from: https://towardsdatascience.com/python-scikit-learn-logistic-regression-classification-eb9c8de8938d [Accessed on November 15th 2020]

Srinivasan, D. (2020) *Why Google Dominates Advertising Markets.* Available from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3500919 [Accessed on November 5th 2020]

Appendix